ON FOUNDATIONS OF PROTECTING COMPUTATIONS

zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften

von der KIT-Fakultät für Informatik
des Karlsruher Instituts für Technologie (KIT)

genehmigte

Dissertation

von

Thomas Agrikola

geboren am 03. April 1991 in Karsruhe

Tag der mündlichen Prüfung: 07. Mai 2021

1. Referent: Prof. Dr. Jörn Müller-Quade
2. Referent: Prof. Dr. Dennis Hofheinz

ON FOUNDATIONS OF PROTECTING COMPUTATIONS

Dissertation by Thomas Agrikola in partial fulfillment of the requirements for the degree of Doctor of Natural Sciences (Dr. rer. nat.) accepted by the KIT Department of Informatics of the Karlsruhe Institute of Technology (KIT)

Day of examination: May 7, 2021
First referee: Prof. Dr. Jörn Müller-Quade, KIT
Second referee: Prof. Dr. Dennis Hofheinz, ETH Zürich

Published by KIT, Karlsruhe, Germany
DOI: 10.5445/IR/1000133798

ABSTRACT

Information technology systems have become indispensable to uphold our way of living, our economy and our safety. Failure of these systems can have devastating effects. Consequently, securing these systems against malicious intentions deserves our utmost attention.

Cryptography provides the necessary foundations for that purpose. In particular, it provides a set of building blocks which allow to secure larger information systems. Furthermore, cryptography develops concepts and techniques towards realizing these building blocks. The *protection of computations* is one invaluable concept for cryptography which paves the way towards realizing a multitude of cryptographic tools. In this thesis, we contribute to this concept of protecting computations in several ways.

PROTECTING COMPUTATIONS OF PROBABILISTIC PROGRAMS. An indistinguishability obfuscator (IO) compiles (deterministic) code such that it becomes provably unintelligible. This can be viewed as the ultimate way to protect (deterministic) computations. Due to very recent research, such obfuscators enjoy plausible candidate constructions.

In certain settings, however, it is necessary to protect *probabilistic* computations. The only known construction of an obfuscator for probabilistic programs is due to Canetti, Lin, Tessaro, and Vaikuntanathan, TCC, 2015 and requires an indistinguishability obfuscator which satisfies extreme security guarantees. We improve this construction and thereby reduce the requirements on the security of the underlying indistinguishability obfuscator.

Agrikola, Couteau, and Hofheinz, PKC, 2020

PROTECTING COMPUTATIONS IN CRYPTOGRAPHIC GROUPS. To facilitate the analysis of building blocks which are based on cryptographic groups, these groups are often overidealized such that computations in the group are protected from the outside. Using such overidealizations allows to prove building blocks secure which are sometimes beyond the reach of standard model techniques. However, these overidealizations are subject to certain impossibility results. Recently, Fuchsbauer, Kiltz, and Loss, CRYPTO, 2018 introduced the algebraic group model (AGM) as a relaxation which is closer to the standard model but in several aspects preserves the power of said overidealizations. However, their model still suffers from implausibilities. We develop a framework which allows to transport several security proofs from the AGM into the standard model, thereby evading the above implausibility results, and instantiate this framework using an indistinguishability obfuscator.

Agrikola, Hofheinz, and Kastner, EUROCRYPT, 2020

PROTECTING COMPUTATIONS USING COMPRESSION. Perfect compression algorithms admit the property that the compressed distribution is truly random leaving no room for any further compression. This property is invaluable for several cryptographic applications such as "honey encryption"

or password-authenticated key exchange. However, perfect compression algorithms only exist for a very small number of distributions. We relax the notion of compression and rigorously study the resulting notion which we call "pseudorandom encodings". As a result, we identify various surprising connections between seemingly unrelated areas of cryptography. Particularly, we derive novel results for adaptively secure multi-party computation which allows for protecting computations in distributed settings. Furthermore, we instantiate the weakest version of pseudorandom encodings which suffices for adaptively secure multi-party computation using an indistinguishability obfuscator.

Agrikola, Couteau, Ishai, Jarecki, and Sahai, TCC, 2020

PUBLICATIONS

[AH18] Thomas Agrikola and Dennis Hofheinz. "Interactively Secure Groups from Obfuscation." In: *Public-Key Cryptography - PKC 2018 - 21st IACR International Conference on Practice and Theory of Public-Key Cryptography, Rio de Janeiro, Brazil, March 25-29, 2018, Proceedings, Part II*. (Work partially included in my master's thesis.) 2018, pp. 341–370. DOI: 10.1007/978-3-319-76581-5_12. EPRINT: https://eprint.iacr.org/2018/010. URL: https://doi.org/10.1007/978-3-319-76581-5_12 (cit. on pp. 9, 91, 92, 110, 111, 115, 116, 120).

[ACH20] Thomas Agrikola, Geoffroy Couteau, and Dennis Hofheinz. "The Usefulness of Sparsifiable Inputs: How to Avoid Subexponential iO." In: *Public-Key Cryptography - PKC 2020 - 23rd IACR International Conference on Practice and Theory of Public-Key Cryptography, Edinburgh, UK, May 4-7, 2020, Proceedings, Part I*. Ed. by Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas. Vol. 12110. Lecture Notes in Computer Science. Springer, 2020, pp. 187–219. DOI: 10.1007/978-3-030-45374-9_7. EPRINT: https://eprint.iacr.org/2018/470. URL: https://doi.org/10.1007/978-3-030-45374-9_7 (cit. on pp. 3–5, 17–19, 41, 46, 71, 81, 245).

[ACI+20] Thomas Agrikola, Geoffroy Couteau, Yuval Ishai, Stanislaw Jarecki, and Amit Sahai. "On Pseudorandom Encodings." In: *Theory of Cryptography - 18th International Conference, TCC 2020, Durham, NC, USA, November 16-19, 2020, Proceedings, Part III*. Ed. by Rafael Pass and Krzysztof Pietrzak. Vol. 12552. Lecture Notes in Computer Science. Springer, 2020, pp. 639–669. DOI: 10.1007/978-3-030-64381-2_23. URL: https://doi.org/10.1007/978-3-030-64381-2_23 (cit. on pp. 4, 5, 12, 15, 17–19, 147, 149, 153, 155, 245).

[AHK20] Thomas Agrikola, Dennis Hofheinz, and Julia Kastner. "On Instantiating the Algebraic Group Model from Falsifiable Assumptions." In: *Advances in Cryptology - EUROCRYPT 2020 - 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10-14, 2020, Proceedings, Part II*. Ed. by Anne Canteaut and Yuval Ishai. Vol. 12106. Lecture Notes in Computer Science. Springer, 2020, pp. 96–126. DOI: 10.1007/978-3-030-45724-2_4. EPRINT: https://eprint.iacr.org/2020/070. URL: https://doi.org/10.1007/978-3-030-45724-2_4 (cit. on pp. 4, 5, 9, 10, 17–19, 87, 89, 91, 130, 245).

We can only see a short distance ahead,
but we can see plenty there that needs to be done.

— Alan Turing [Tur50]

ACKNOWLEDGMENTS

First and foremost, I would like to thank my advisor Dennis Hofheinz for his support and for awakening my interest in cryptography in the first place. Your technical intuition for cryptography is enviable. I also would like to thank Jörn Müller-Quade for making it possible to finish my PhD at KIT when Dennis decided to leave for ETH Zürich.

Furthermore, I would like to thank my co-authors Geoffroy Couteau, Yuval Ishai, Stanisław Jarecki, Julia Kastner and Amit Sahai for many inspiring discussions, for your ability to make hard problems seem easy and for your trust. It was an honor and a pleasure to work with you.

I cannot emphasize enough how grateful I am to Geoffroy. You were like a second advisor to me and without your support during a rather frustrating time, I might not be where I am now. Thank you!

Special thanks also goes to Michael Klooß and Sven Maier for thoroughly proofreading this thesis.

Also, I would like to thank all my former and present colleagues for the great working atmosphere and for making my time at the institute so enjoyable. In particular, I would like to thank Akin Ünal for decorating your office beyond comparison; Björn Kaidel for encouraging me to major in cryptography; Bogdan Ursu for the countless hours of trying to best me at table tennis; Brandon Broadnax for a great time at the conference in Brazil and for always decimating my cough drops; Geoffroy Couteau for many fun and entertaining discussions about everything; Jessica Koch for the fun time we had when co-organizing the practice part for the security lecture together; Julia Hesse for Elin; Julia Kastner for sharing your limitless knowledge about squirrels; Lisa Kohl for fun hangman games during exam supervisions; Michael Klooß for your unparalleled love for formalisms; Sven Maier for an unending supply of Doctor Who DVDs; Valerie Fetzer for a great canoeing trip.

Finally, I would like to thank my family for their encouragement and support. Furthermore, I would like to thank my better half Dani for all her support and patience during my PhD. Together, we can overcome anything.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

$\mathcal{BPP}$	Bounded probability polynomial time. 21, 35
$\mathcal{NC}$	Nick's class. 2
$\mathcal{NP}$	Non-deterministic polynomial time. 1–3, 21, 25, 35, 61, 205, 211
$\mathcal{P}$	Deterministic polynomial time. 1, 2, 21, 25
$\mathcal{UP}$	Unambiguous polynomial-time. 94
$\mathcal{P}/\mathsf{poly}$	Non-uniform polynomial time. 21
AGM	Algebraic group model. iii, 9–12, 87–94, 109, 111, 245
AMPC	Fully adaptively secure multi-party computation. 221, 224, 226
AT	Anonymous transfer. 246

BLS Boneh-Lynn-Shacham signature scheme. 12, 89

CRS Common reference string. xv, 15, 46, 63, 95, 149, 150, 152, 154, 156, 158, 161–163, 171, 172, 178, 181, 187, 194, 197, 199, 204, 205, 207, 212, 221–224, 226, 227, 231, 237–239

DDH Decisional Diffie-Hellman assumption. 42, 43, 45, 46, 48, 49, 80, 95, 129, 240

DE Deniable encryption. 233

DIO Differing-inputs obfuscation. 27

DL Discrete logarithm assumption. 93, 129, 130, 134

DLIN Decisional linear assumption. 96

DPIO Doubly-probabilistic indistinguishability obfuscation. xii, 51–56, 59–62, 71, 73–75, 81, 245

DTE Distribution transforming encoder. xv, 156, 158, 230, 231

EDDH Exponential decisional Diffie-Hellman assumption. 48, 49

ELF Extremely lossy function. 7, 8, 42, 44–49, 61, 63, 66, 80, 81, 238–240

EOWF Extractable one-way function. xii, 31–33, 88, 152–154, 200, 204, 205, 211, 212, 216

FHE Fully homomorphic encryption. 34, 42, 43, 45, 46, 80, 81, 97

GEOWF Generalized extractable one-way function. xiv, 153, 200–202, 204, 211

GGM Generic group model. 8, 9, 87, 93

GKEA Generalized knowledge-of-exponent assumption. 109

HE Honey encryption. 230

IACR International Association of Cryptologic Research. 4, 17

IND-CCA1 Indistinguishability under (non-adaptive) chosen-ciphertext attack. 46, 245

IND-CCA2 Indistinguishability under chosen-ciphertext attack. 89, 129–131

IND-CPA Indistinguishability under chosen-plaintext attack. xv, 3, 34, 73–76, 78–80, 129, 213, 233–237

IO Indistinguishability obfuscation. iii, xiv, 2–8, 10–12, 15, 17, 18, 21, 25–27, 29, 33, 41–46, 60, 71, 73, 80, 81, 89, 91, 92, 97, 99, 149, 152–154, 158, 159, 187, 211, 215, 231, 245

ISH Invertible sampling hypothesis. 151, 152

KDM Key-dependent message attack. 42, 45, 46, 73

KEA Knowledge-of-exponent assumption. 30–33

KEM Key-encapsulation mechanism. 129

KLSTS Key-less stegosystem. 231

LHE Leveled homomorphic encryption. xii, 8, 45, 73–75, 80, 81

LPN Learning-pairity-with-noise assumption. 2, 245

LWE Learning-with-errors assumption. 2, 3, 36, 43, 153, 158, 159, 187, 200, 245

MPC Multi-party computation. 3, 16, 17, 149, 151–155, 171, 187, 221, 222, 229, 237, 241, 246

NCE Non-committing encryption. 237

NIWI Non-interactive witness-indistinguishable proof system. 95, 96, 115, 117, 212

NIZK Non-interactive zero-knowledge proof system. 22, 35, 36, 44, 59–61, 63, 96, 117, 153, 199, 200, 204, 205, 210–212

OT Oblivious transfer. 3, 224

OWF One-way function. 23, 33, 97, 153

PAKE Password-authenticated key exchange. 150, 158

PIO Probabilistic indistinguishability obfuscation. xii, xiii, 5–7, 27–29, 42–46, 60, 80, 99–102, 104, 105, 116, 117, 127

PKE Public-key encryption. 33, 46, 97, 153, 221, 233, 234, 237

PKSTS Public-key stegosystem. 231

PPAD Polynomial parity argument (directed). 3

PPRF Puncturable pseudorandom function. xii, 23, 24, 47, 102–105

PPT Probabilistic polynomial time. 19, 21–26, 28, 31–36, 47, 52, 53, 55, 56, 62, 63, 65, 66, 68–71, 73, 76–80, 93–97, 101, 102, 105, 106, 112, 113, 116, 120–124, 126, 130, 148, 151, 154, 159–161, 163, 165–185, 187, 197, 200, 201, 205, 207, 208, 210, 212, 214, 216, 217, 219, 221–229, 231–234, 236, 237, 239–241

PRE Pseudorandom encoding. 149

PREH Pseudorandom encoding hypothesis. 14, 148, 168

PRF Pseudorandom function. xii, 3, 23, 24, 29, 43, 45, 47, 59–61, 64, 69–71, 94, 100, 102–105, 109, 212, 214, 215

PRG Pseudorandom generator. 2, 167, 192, 212, 214

QRA Quadratic residuosity assumption. 153, 157, 163, 199

ROM Random oracle model. 10, 88, 90, 91

SKE Symmetric-key encryption. 33, 230

SMP Subset membership problem. 94, 95

SNARG Succinct non-interactive arguments. 27

SS-CCA Chosen-covertext attack. 231

SSB Somewhere statistically binding. 3

SXDH Symmetric external Diffie-Hellman assumption. 96

UC Universal composability framework. 223, 224, 226

UPRE Universal proxy re-encryption. 46

URC Uniformly random CRS. 171, 181, 216, 219, 229

VBB Virtual black-box obfuscation. 25

VLF Very lossy function. 238–241

1 INTRODUCTION

WHY CRYPTOGRAPHY? In the current information age, our economy is primarily built on information technology. This applies for our society as well. Access to and distribution of information has become an essential part of our daily lives.

Many critical infrastructures rely on information technology systems. This includes, for instance, control systems for nuclear power plants, military drones, smart grids, online payment systems, internal databases, medical records, personal correspondence and social media. Failure of these systems has severe consequences ranging from damage of social lives and reputations, over damage to economic infrastructures, to endangerment of human lives. In fact, on account of this dependency on information technology, cyberwarfare, cyberterrorism and cybercrime have become as devastating as their physical counterparts.

Consequently, the necessity to ensure security of information systems has become pervasive.

WHAT IS CRYPTOGRAPHY? Cryptography provides building blocks which are capable of creating an *asymmetry* in the computational costs between honest users and malicious entities. These building blocks can then be used to ensure security in larger information systems. One primary focus of cryptography is to provide *rigorous mathematical proofs* of such building blocks' properties and guarantees.

One widely used approach towards achieving such an asymmetry is to resort to computational complexity theory. This approach is referred to as asymptotic security. The philosophy behind asymptotic security is to be able to scale the security of a building block to withstand *arbitrarily powerful* adversaries, while the computational cost for honest users remains affordable. That is, any supposed computational adversarial power, even magnitudes of whole computing centers, can be outmatched by computationally limited users.

For most cryptographic building blocks, the task of breaking them is in $\mathcal{NP}$, hence non-deterministically solvable in polynomial time (since solution candidates can be verified in polynomial time). Consequently, given our current understanding of computational complexity, asymptotic security cannot be achieved unconditionally but requires computational hardness assumptions. Clearly, $\mathcal{P} \neq \mathcal{NP}$ is a necessary assumption.

However, the assumption $\mathcal{P} \neq \mathcal{NP}$ is not sufficient for cryptography since the definition of $\mathcal{NP}$ only considers hardness in the worst case. In the context of cryptography, not only need there be hard problems, but furthermore sampling hard instances of these problems must be efficient since this task is performed by an honest user. The initial foundational study of hardness-on-the-average is due to Levin [Lev86].

SOURCES OF HARDNESS. One central goal of cryptography is to identify and analyze potential sources of computational hardness. Number theory gives rise to several computational hardness assumptions such as (variants of) the factoring assumption [RSA78] and assumptions in cyclic groups [DH76]. Some time later, codes [BFKL94] and lattices [Ajt96] were discovered to yield new sources of computational hardness.

An alternative road towards secure cryptographic building blocks is to base their security on *generic* hardness assumptions which can be instantiated from several sources of hardness. The most fundamental generic hardness assumption is the existence of one-way functions [DH76], i. e., functions which can efficiently be evaluated but are hard to invert. To justify security based on generic assumptions which do not admit any inherent algebraic structure (such as the aforementioned sources of hardness) is rather appealing from a theoretical perspective. Furthermore, if one particular source of hardness was to fall victim to modern cryptanalysis, there remain several other sources that can equally be used.

In the recent past, another generic hardness assumption has been conceived – the assumption that computations within certain programs are protected. In the following, we shed light on several angles of protecting computations.

CODE OBFUSCATION. Compiling program code such that it becomes *provably* unintelligible can be seen as the ultimate way to protect computations.[1] The concept of code obfuscation was first formalized in [Had00; BGI+12]. An obfuscator is a compiler which preserves the functionality of programs but makes the original code unintelligible. In search of the "right" definition of unintelligibility, the notion of indistinguishability obfuscation (IO) emerged as the "best possible" [GR07], achievable and useful [GGH+13] notion. An indistinguishability obfuscator guarantees that the obfuscations of two programs (represented as Boolean circuits) which compute the same functionality are indistinguishable from each other. Note that since IO can exist even if $\mathcal{P} = \mathcal{NP}$, an additional mild assumption is usually necessary to instantiate a cryptographic building block from IO.

[1] *This should not be confused with practical code obfuscation used in the realm of software engineering which does not provide any provable guarantees, e. g., [CTL98a; CTL98b].*

Starting from the line of work [Lin16; LV16; LT17; Lin17], there has been substantial progress towards instantiating IO from (falsifiable) standard assumptions [AJL+19; Agr19; JLMS19; BDGM20a; GJLS20; JLS20; GP20; BDGM20b]. Most remarkably, Jain, Lin, and Sahai [JLS20] introduce the first instantiation of IO based solely on well-founded assumptions, namely pairing-friendly cryptographic groups, LWE, LPN and PRGs in $\mathcal{NC}^0$. Another remarkable candidate is due to Brakerski, Döttling, Garg, and Malavolta [BDGM20b] which is based only (a slightly non-standard variant of) circularly secure LWE.

Another line of work seeks to remove pairing based assumptions from the above works [Agr19; AP20]. Very recently, [WW20], building upon [BDGM20a], instantiate IO from lattice-based assumptions in conjunction with a novel but falsifiable oblivious sampling assumption.

Furthermore, there has been significant progress towards instantiating IO from graded multilinear maps (graded encoding schemes) [GGH15; BMSZ16; GMM+16; CVW18; CHVW19]. These current candidates for graded encoding schemes only provide heuristic security (that is, they are proven secure in idealized models) but withstood intense cryptanalysis [CHL+15; MSZ16; CLLT16; ADGM17; CLLT17; CGH17; CVW18].

Another incomparable line of work base IO directly on assumptions about tensor products [GJK18] and affine determinant programs [BIJ+20].

Protecting computations by making code unintelligible emerged as an extremely powerful tool unveiling undreamt-of possibilities. Following the positive result of [GGH+13], a wide variety of applications has been proposed, e. g., functional encryption [GGH+13], deniable encryption [SW14], fully homomorphic encryption [CLTV15], two-round secure multi-party computation [GGHR14], multi-party key exchange [BZ14], secret sharing for $\mathcal{NP}$ [KNY14], IO for Turing machines [KLW15], succinct randomized encodings [BGL+15], constant-round concurrent zero-knowledge [CLP15], reductions from the hardness of PPAD to the hardness of cryptographic problems [BPR15; GPS16], trapdoor permutations [BPW16; CL18], non-interactive zero-knowledge proof systems (in the common random string model) and non-interactive witness-indistinguishable proof systems [BP15a]. This includes applications that seemed beyond our reach for a long time and became feasible thanks to IO.

Not only does IO yield intriguing new applications, but it also triggers the development of new techniques. These techniques prove useful even in settings without IO. The core technique which was developed for IO is "puncturing" [SW14]. This motivated the study and use of puncturable pseudorandom functions (PRFs). Puncturing proved useful in further areas and puncturable PRFs are at the heart of recent constructions like homomorphic secret sharing [BGI17], oblivious transfer with small communication [BCG+19a], and pseudorandom correlation generators [BCG+19b]. Further, IO motivated the design of "somewhere statistically binding (SSB) hashing" [HW15; OPWW15] as an abstraction of useful properties for the analysis of IO-based constructions. This notion of SSB hashing has inspired recent results like laconic OT [CDG+17] and trapdoor hash functions [DGI+19] which both have significant applications for MPC. Due to [GGG+14], IO can be used to achieve two-round MPC. This inspired the notion of interactive garbling schemes [FJNT15] which led to the result that two-round MPC can be instantiated from OT [GS18; BL18a]. Moreover, IO techniques proved useful for identity-based encryption [DG17], verifiable random functions [Bit17; GHKW17] and the Fiat-Shamir paradigm and correlation intractability allowing to emulate obfuscation-like properties from key-dependent message security [CCRR18; HL18; CCH+19; PS19]. Furthermore, IO motivated the study of multilinear maps. The candidate construction from [GGH15] can indeed be proven secure in certain settings under the LWE assumption paving the way for novel LWE-based constructions such as privately constrained PRFs [CC17], compute-and-compare obfuscation [WZ17] and separations between circular security and IND-CPA security for secret-key encryption [GKW17].

Thereby, the study of IO helps to improve our understanding of cryptography.

OVERVIEW OF CONTRIBUTIONS. In the following, we provide a high-level overview of the contributions of this thesis, see Figure 1.1.

We improve concepts regarding the protection of computations in several ways. In [ACH20], we improve constructions in the context of probabilistic obfuscation – a variant of indistinguishability obfuscation for randomized programs. To be specific, previous constructions of probabilistic IO required

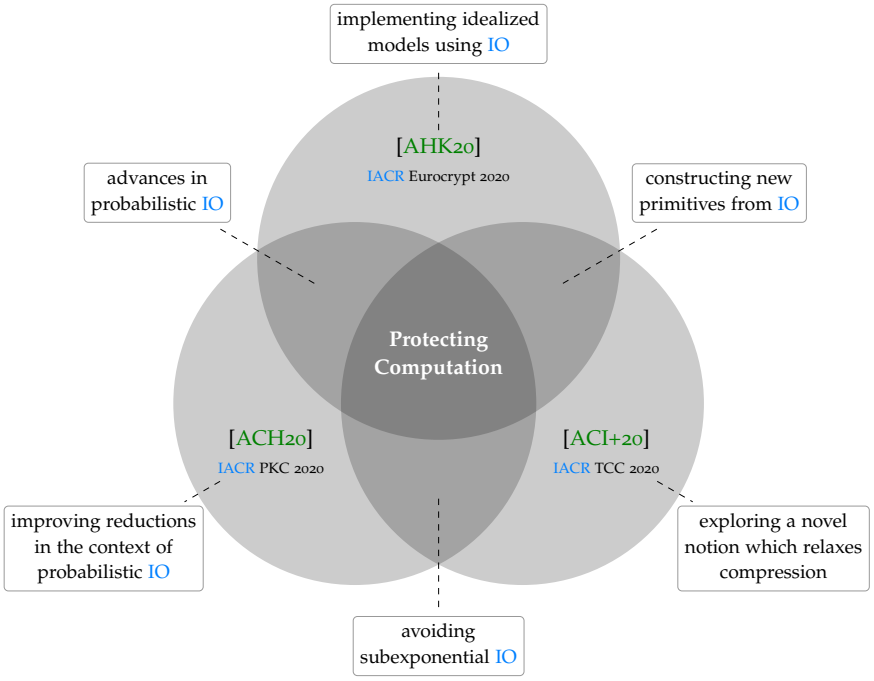


Figure 1.1: A high-level overview of the contributions of this thesis.

[2] *Subexponentially secure IO provides guarantees even against subexponential-time adversaries. Cryptographic hardness assumptions classically only require security against polynomial-time adversaries which is a much weaker requirement.*

subexponentially secure[2] IO, a very strong assumption. We manage to reduce this requirement to standard polynomially secure IO. In [ACI+20], we relax the notion of perfect compression to harness the useful property that compressed samples appear to be uniform randomness for a broader class of distributions. We derive a notion we call "pseudorandom encodings" which we prove to exist for all efficiently samplable distributions assuming IO. Further, our study of pseudorandom encodings reveals interesting and unexpected relations to secure multi-party computation which allows to protect computations in distributed systems. In [AHK20], we use IO to instantiate an idealized model in the context of cryptographic groups which protects – or shields – computations in the group. While these contributions are centered around the protection of computations, this is not their only common ground.

[ACH20; AHK20] both make progress in the field of probabilistic IO. [ACH20] reduces the necessary hardness assumptions and [AHK20] improves the correctness guarantee offered by probabilistic IO. More precisely, previous notions of probabilistic IO information-theoretically leaked whether an output was produced via the original or the obfuscated program while our novel notion of probabilistic IO conceals even this information. Hence, we make progress in the context of protecting randomized computations.

[ACH20] avoids assuming subexponential IO. Surprisingly, [ACI+20] also avoids this assumption in a completely different context: adaptively secure multi-party computation with sublinear communication complexity [CsW19]. Thereby, we make progress towards realizing protection of computations based on weaker and more reliable hardness assumptions.

Furthermore, we make progress in deriving novel building blocks from the assumption that certain computations are protected. In [ACI+20], we construct a novel primitive (our relaxation of compression) from IO. This is also true for [AHK20], where we construct a novel building block from IO

which can be used to transport proofs from an idealized group model to the standard group model.

In the following, we elaborate on these results in more detail.[3]

[3] *Note that parts of the following are loosely inspired by the introductions of the underlying publications [ACH20; AHK20; ACI+20].*

1.1 PROTECTING COMPUTATIONS OF PROBABILISTIC PROGRAMS

Presently, almost all conceivable cryptographic building blocks can be instantiated from IO (and mild generic assumptions).[4] However, some building blocks currently require *subexponentially* secure IO, i. e., require security even against subexponential-time adversaries with time complexity in $\mathcal{O}(2^{\lambda^\epsilon})$ (for some $0 \leqslant \epsilon < 1$).[5] This is a much stronger requirement compared to the standard notion of security which covers polynomial-time adversaries.

[4] *Except for the notable examples of collision resistant hash functions or fully-homomorphic signatures.*

[5] *This definition of subexponential security is a slight simplification which suffices for this overview.*

There are several promising roads towards achieving IO. The approaches based on functional encryption start from subexponential assumptions and directly yield subexponential IO "for free". This includes the recent breakthrough results [BDGM20a; GJLS20; JLS20; GP20; BDGM20b] outlined above. However, this does not apply to the approaches using graded encoding schemes [GGH15; BMSZ16; GMM+16; CVW18; CHVW19] and the incomparable approaches from tensor products [GJK18] or affine determinant programs [BIJ+20]. For these approaches, security against polynomial-time adversaries and security against subexponential-time adversaries does make a significant difference. Furthermore, constructing a building block from polynomial IO relates the security of that building block more tightly to IO. This can give rise to more insightful techniques and help to improve our understanding of that building block. Thus, it is desirable to avoid subexponentially secure IO.

RELATED WORK ON AVOIDING SUBEXPONENTIAL IO. Avoiding subexponentially secure IO has been studied in [GPS16; GPSZ17; LZ17; KLMR18]. These works follow the approach to replace (subexponentially secure) IO with polynomially secure functional encryption. This strong variant of public-key encryption allows to implement several IO-based proof techniques without relying on guarantees against subexponential-time adversaries. However, this strategy requires the building block in question to meet certain structural conditions. We seek to provide a more general solution.

PROBABILISTIC OBFUSCATION. A certain class of building blocks requires to protect *probabilistic* computations. The notion of probabilistic indistinguishability obfuscation (pIO) was put forth by Canetti, Lin, Tessaro, and Vaikuntanathan [CLTV15] and requires subexponentially secure IO. On a high level, pIO not only protects the program code, but additionally protects the randomness which is used to evaluate the randomized program from a user of the program. Syntactically, a probabilistic indistinguishability obfuscator compiles a randomized circuit into a deterministic circuit which has "almost" the same functionality. Recall that standard IO guarantees indistinguishability of obfuscated programs, provided that these programs behave identically on all inputs. For a useful definition, this condition of identical behavior needs to be relaxed such that pIO is applicable for, e. g.,

programs which produce identical output distributions. Informally, a probabilistic indistinguishability obfuscator guarantees that the obfuscations of two randomized programs are indistinguishable, provided that these programs induce suitably close output distributions. Currently, the only known construction of pIO applies standard IO on a de-randomized version of the original program. De-randomized programs derive their randomness deterministically by applying a pseudorandom function[6] F on their input, that is,

$$\mathsf{piO}(C) := \mathsf{iO}(\overline{C}), \text{ where } \overline{C}(x) := C(x; \mathsf{F}(x)).$$

[6] *A pseudorandom function is a function which is indistinguishable from a truly random function.*

The reason why polynomially secure IO is insufficient for pIO is best explained by example. Consider the two randomized programs C_0 and C_1, where C_b on input of $x \in \{0,1\}^\lambda$ and (internal) randomness $r \in \{0,1\}^\lambda$, outputs $x \oplus r \oplus b^\lambda$ (for $b \in \{0,1\}$). Clearly, these programs both induce the uniform distribution over $\{0,1\}^\lambda$. Hence, pIO should guarantee indistinguishability between $\mathsf{piO}(C_0) = \mathsf{iO}(\overline{C_0})$ and $\mathsf{piO}(C_1) = \mathsf{iO}(\overline{C_1})$. However, IO does not provide any guarantees for circuits which do not behave fully identically. In fact, there is not even one input x such that the de-randomized circuits $\overline{C_0}$ and $\overline{C_1}$ produce the same output.

The only known strategy to deal with this issue is to prove that $\mathsf{iO}(\overline{C_0})$ and $\mathsf{iO}(\overline{C_1})$ are indistinguishable *for every single input* x (applying the security property of IO every time). Since the number of inputs is exponential, this proof technique inherently requires (sub)exponential security of IO.

We aim at realizing a sufficient protection of randomized computations without having to rely on subexponentially secure IO.

DOUBLY-PROBABILISTIC IO. For the purpose of avoiding subexponential IO, a natural solution would be to reduce the number of inputs for the probabilistic circuit. However, obfuscation with only polynomially many inputs is trivially solvable by enumerating each input-output-pair and can thus not be useful towards constructing complex cryptographic building blocks.

We observe that in many cases, it suffices to consider probabilistic circuits expecting inputs from a small number of efficiently samplable distributions. To exemplify, consider the two distributions induced by encryption of a single bit. In many applications of pIO, the obfuscated circuit expects only inputs which originate from such distributions. Towards reducing the number of meaningful inputs which need to be considered in the security proof, we restrict the correctness guarantee provided by pIO to only hold for "well-generated" inputs. To exclude all "non-well-generated" inputs from the security proof, we certify each input in the domain of an input sampler with a non-interactive zero-knowledge proof and define the probabilistic circuit to reject all uncertified inputs.[7] However, while the number of distributions is small, the support of each such distribution is still exponential.

[7] *This approach only certifies membership, no distributional property.*

The restricted correctness property lays the foundation to an orthogonal way to reduce the amount of well-generated inputs. Namely, suppose we could narrow the randomness space of the input samplers to a polynomial amount, the amount of well-generated inputs necessarily shrinks to a polynomial. However, shrinking the randomness space from an exponential size to a polynomial size is far from trivial. Consider the example of bit-encryption distributions from above. For these distributions, it is essential to use

good randomness to ensure secrecy of the encrypted bit. If the randomness space for encryption is only polynomial, exhaustive search can recover the encrypted bit.

Contrary to the above intuition, it is indeed possible to sparsify the randomness space to a polynomial size using a special tool called *extremely lossy functions* introduced by Zhandry [Zha16]. Extremely lossy functions (ELFs) are functions which can be set up in two computationally indistinguishable modes – an injective mode and an extremely lossy mode. In injective mode, the range of the ELF is exponential whereas in extremely lossy mode, the range of the ELF merely comprises a polynomial quantity. Such a notion requires a slightly non-black-box use of adversary. More precisely, such an indistinguishability cannot hold against arbitrary polynomial-time adversaries since finding collisions requires only polynomial resources. Instead, the size of the range in extremely lossy mode needs to be sufficiently large to fool the particular adversary in question.

We use extremely lossy functions to pre-process the randomness for the input sampler, see also Figure 1.2 for an overview. In injective mode, the original distribution is not affected. During the proof, we can unnoticeably (for the particular adversary) switch the ELF into extremely lossy mode, thereby sparsifying the randomness space of the input sampler and hence the amount of well-generated inputs.

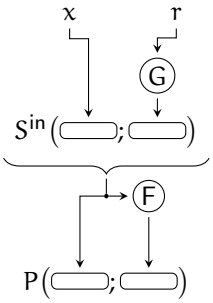


Figure 1.2: Overview of the strategy to reduce the amount of hybrids to a polynomial. The randomized program P receives inputs from some input distribution S^{in} and derives its randomness using a pseudorandom function F on this input. The input distribution S^{in} receives an input x from a small input space and a random tape r (from an exponentially large randomness space). Pre-processing r with the extremely lossy function G allows to sparsify the amount of possible inputs to P to a polynomial amount.

Using the strategy above, we are able to use standard polynomially secure IO by "pushing" the subexponential loss to a much more standard and better-understood assumption on cyclic groups which is necessary to instantiate the extremely lossy function. We provide a general framework called "doubly-probabilistic IO" which can be used to avoid subexponentially secure IO in many pIO-based applications without making any assumptions on the structure of the obfuscated programs. We only need to require that inputs for the obfuscated circuit originate from a small number of efficiently samplable distributions.

APPLICATIONS. We are able to apply doubly-probabilistic IO, for instance, in the context of leveled homomorphic encryption (LHE). LHE allows to evaluate fixed-depth circuits on the contents of ciphertexts.

The LHE construction due to Canetti et al. [CLTV15] requires to obfuscate a probabilistic program which expects two ciphertexts as input, decrypts them, evaluates say the logical NAND on the plaintexts, and re-encrypts the result. Clearly, this program expects two inputs which both come from one of two efficiently samplable distributions: the ciphertext distribution of the plaintext 0 or the ciphertext distribution of the plaintext 1. Hence, by using doubly-probabilistic IO, we are able to push the subexponential assumptions for LHE away from IO to ELFs which can be based on well-established and well-analyzed assumptions.

Thus, our notion of doubly-probabilistic IO provides a general framework for reducing the necessary assumptions for protecting probabilistic computations. We refer the reader to Part I of this thesis for more details.

SUBSEQUENT WORK. Our notion of doubly-probabilistic IO has already found further application. Döttling and Nishimaki [DN18] are able to avoid subexponentially secure IO in the following context. [DN18] consider the task of converting ciphertexts produced with *any* public-key encryption scheme into re-randomized ciphertexts under *any possibly different* public-key encryption scheme. They call this notion universal proxy re-encryption and instantiate it using probabilistic IO. Furthermore, they observe that using our doubly-probabilistic IO, they are able to avoid subexponentially secure IO.

1.2 PROTECTING COMPUTATIONS IN CRYPTOGRAPHIC GROUPS

Since the seminal work of Diffie and Hellman [DH76], certain cyclic groups have emerged as a major source of computational hardness giving rise to a great variety of cryptographic building blocks such as public-key encryption and digital signatures [ElG85], non-committing encryption [Bea97], identity-based encryption [BF01], non-interactive zero-knowledge proof systems [GOS12], and many more.

Protecting computations is a central concept in the context of cryptographic groups. Building blocks or novel hardness assumptions which are based on cryptographic groups are commonly analyzed in an idealized group model. This idealized model assumes that the computations in the group are "protected" in the sense that an adversary has no control over the group itself, but is granted access to a restricted interface providing only generic group operations. This model is called the generic group model (GGM) and was formalized by Shoup [Sho97] and Maurer [Mau05]. An adversary who only uses this generic interface is called a generic adversary. Given a security proof in the GGM, no generic adversary can succeed in breaking the proven security properties. In other words, in order to break a generically proven security property in some concrete group, every adversary is required to exploit the concrete encodings of group elements in that group in a nontrivial way. Hence, the GGM provides a sanity check for the security of building blocks or novel assumptions. However, there exist building blocks which

can be proven secure in the GGM but are insecure when instantiated with *any* concrete group [Den02]. Hence, generic groups cannot exist. In my master's thesis which led to the publication [AH18], we demonstrate that fundamental proof strategies developed for the GGM can be transported into the standard model at the expense of uniqueness of group element encodings. However, due to this lack of unique group element encodings and the inability to extract a unique bitstring from group element encodings, the applicability of the resulting group is very limited. Without the ability to extract a bitstring which is unique for every encoding of the same group element, it is impossible to extract, e. g., a unique encryption key from a group element. This result is not part of this thesis, but [AHK20], which is part of this thesis, constitutes follow-up work.

ALGEBRAIC GROUPS. The algebraic group model [FKL18] is a relaxation of the generic group model conceived to avoid the above mentioned impossibility results while preserving the power and generality of the GGM. While the GGM only considers the class of generic adversaries, the AGM considers the broader class of so-called algebraic adversaries. In contrast to generic adversaries, algebraic adversaries can make arbitrary use of group element encodings but must explain each output group element in terms of their input group elements. More precisely, if an algebraic adversary outputs a group element h given the group elements $g_1, \ldots, g_\ell$, the adversary must additionally provide a list of exponents $z_1, \ldots, z_\ell$, such that $h = \prod_{i=1}^{\ell} g_i^{z_i}$. This formalization of algebraic adversaries is equivalent to the more standard definition that for every adversary who outputs a group element, there exists an efficient extractor which is able to derive the exponents $z_1, \ldots, z_\ell$ as above. Every generic adversary is algebraic, since a generic adversary needs to use a generic interface of oracles to obtain new group elements.

Hence, while the AGM does not protect computations to the same extent as the GGM, it still offers a meaningful protection of computations in the group. Namely, even though an adversary can derive group elements in an arbitrary way disregarding the actual group operation, ultimately, he has to be able to explain his doing in terms of the group operation. Thus, the AGM protects the computations in the group by nullifying any advantage in deriving group elements in any other way than using the group operation directly.

INHERENT OBSTACLES. The AGM is equivalent to a very strong form of the knowledge-of-exponent assumption [Dam92; WS07; KP19]. Knowledge-type assumptions basically state that if an adversary can produce an output, this adversary *must know* how he computed this output. To reason about adversarial knowledge is a common concept in cryptography. Adversarial knowledge of information is modeled by the ability to algorithmically extract this information from the adversary's code using an efficient extractor. Consequently, while escaping the impossibility results of the GGM, the AGM nevertheless suffers from several inherent obstacles.

Cryptographic hardness assumptions can be classified into falsifiable and non-falsifiable ones based on the complexity of falsifying them. An assumption is falsifiable if an observation can show it to be false. More precisely, a cryptographic assumption is falsifiable if the assumption can be formalized as an interaction between an *efficient* challenger and an adversary such that

the challenger can efficiently provide the adversary with a challenge and is able to efficiently verify the adversary's solution attempt. This classification was put forth by Naor [Nao03]. Lacking the property of efficient falsifiability renders assessing the plausibility of assumptions extremely difficult. Hence, it is (arguably) undesirable to use knowledge-type assumptions (such as the AGM) in cryptography.

Moreover, the AGM suffers from a further obstacle. Certain knowledge-type assumptions such as the AGM which allow an additional unstructured input conflict with IO [BCPR16]. Recall that every adversary in the AGM must explain his output elements as group operations in terms of his input elements. Suppose an adversary receives an obfuscated program as additional input. Suppose further that this obfuscated program, on input of a group element g, computes $g^{F(g)}$, where F is a pseudorandom function. An adversary who receives as input a group element g, evaluates this obfuscated program on g and outputs the result, is certainly unable to explain his output as algebraic operations in terms of his input group element, and can hence not be algebraic as per [FKL18].

Therefore, it is impossible to instantiate the AGM from IO. Instead, we aim at an instantiation of a relaxation of the AGM which preserves the generality of the AGM and can be based solely on falsifiable assumptions.[8] Hence, we tackle the following intriguing question:

To what extent can knowledge-type properties be simulated while keeping all assumptions purely falsifiable?

[8] *The assumption that IO exists is falsifiable in the sense of Naor.*

IMPLEMENTING IDEALIZED MODELS. The use of idealized models which protect computations by shielding it from adversaries is a popular paradigm in cryptography. Aside from cryptographic groups, building blocks which use cryptographic hash functions are often analyzed in an idealized model – the random oracle model (ROM). This model was introduced by Bellare and Rogaway in [BR93] but was implicitly already used in [FS87; GGM84b; GGM86]. In the random oracle model, cryptographic hash functions are modeled as oracles, similar in spirit to the generic group model due to Shoup [Sho97]. Building blocks which come with a proof of security in the ROM are often considerably more efficient compared to building blocks which are provably secure in the standard model.

Replacing idealized models with concrete standard-model implementations is an intriguing problem in cryptography. Hofheinz and Kiltz [HK12] follow this aim by introducing the notion of programmable hash functions. Programmable hash functions allow to transport certain proofs from the random oracle model into the standard model. Following [HK12], a line of work [FHPS13; HSW13; HSW14] transported proofs from the random oracle model to the standard model using multilinear maps or IO. Our results follow this endeavor in the context of cryptographic groups.

INSTANTIATING THE AGM FROM IO. In Part II of this thesis, which is based on the publication [AHK20], we develop a suitable relaxation of the AGM and provide an instantiation based on probabilistic IO and further mild and falsifiable assumptions.

AGM-based proofs heavily rely on the ability to extract a decomposition of the adversarially produced group elements. However, since we strive to

avoid knowledge-type assumptions, extracting this information from the adversary's code is not a viable option. Hence, an alternative extraction mechanism needs to be conceived. We seek to extract a decomposition of group elements *from the group elements themselves*. This type of extraction must exclusively be available during a security proof and must be beyond reach for any user of the group – this includes honest users as well as malicious entities. Hence, we define a special trapdoor which is associated to the public parameters of the group to be necessary to extract the decomposition. This notion of extraction entirely avoids knowledge-type assumptions.

ALGEBRAIC WRAPPER. Albrecht, Farshim, Hofheinz, Larraia, and Paterson [AFH+16] lay the foundations for the construction of groups with non-unique encodings from IO. Groups with non-unique encodings allow each group element to carry auxiliary information inside its representation. We utilize the ability of the [AFH+16] framework to equip any given base group with non-unique encodings. The resulting group virtually "wraps" the group elements of the base group while preserving the original group structure and hardness guarantees. We exploit the thus achieved non-uniqueness of encodings to let each group element carry an *encrypted* decomposition of itself relative to some a priori fixed set of group elements. We refer to this fixed set of group elements as "basis".

Hence, any valid group element is bound to contain a decomposition of itself. See Figure 1.3 for an overview of wrapped group element representations. In particular, this must be true for any *adversarially produced* group element. Consequently, if the corresponding decryption key is known, this decomposition can be extracted from any such group element representation. Intuitively, this forces any adversary to be *algebraic* in a similar sense as in the AGM. Thus, we denote this notion as "algebraic wrapper".

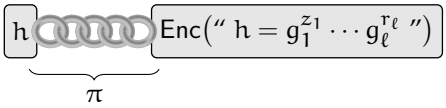


Figure 1.3: Description of group element representations in the algebraic wrapper. Each group element h from the base group is accompanied by a ciphertext containing a decomposition of h with respect to the basis $\{g_1, \ldots, g_\ell\}$. To ensure consistency between h and the encrypted decomposition, a non-interactive zero-knowledge proof π accompanies the representation forcing the encrypted decomposition to match the group element h.

SIMULATING PROOFS FROM THE AGM. For the purpose of proving some building block secure in the AGM, we need to reduce the hardness of some conjectured-to-be-hard problem in the group (e. g., finding discrete logarithms[9]) to the task of breaking that building block. More precisely, an AGM-based proof transforms any algebraic adversary on the building block into a solver for discrete logarithms in the group. In order to simulate this strategy in the standard model, it is essential to be able to deal with arbitrary (not necessarily algebraic) adversaries. In other words, we need to force an arbitrary adversary to provide a decomposition of his output group elements. The ability to extract the decomposition can be ensured by the algebraic wrapper as follows. The reduction (i. e., the discrete logarithm

[9] *The discrete logarithm* $\mathrm{dlog}_g(h)$ *of* h *to the basis* g *is an integer* x *such that* $g^x = h$.

solver) receives a discrete logarithm challenge "$\mathit{dlog}_g(h) = \square$" and sets up parameters for the algebraic wrapper with respect to the given base group. Hence, the reduction is able to perform extraction since the trapdoor (i. e., the decryption key) can be generated alongside the wrapper parameters. The reduction then uses the wrapper group to simulate the security game for the adversary on the building block – suitably embedding the discrete logarithm challenge such that a successful breach of the building block's security allows to extract information about the discrete logarithm challenge.

Generally, using the set $\{g, h\}$ as basis is a good choice. This allows the reduction to generate the group element wrapping h (from the discrete logarithm challenge) since producing group elements requires to know their decomposition with respect to the basis.

LIMITATIONS. Group element representations can only hide the decomposition they carry computationally, i. e., against computationally bounded adversaries. information-theoretically, this decomposition cannot be hidden.[10] However, some AGM proofs require that group element representations information-theoretically look the same, independently of how they were derived (or, in our, case which information they carry). These proofs cannot be simulated using the algebraic wrapper. This includes the tight security proof of the Boneh-Lynn-Shacham (BLS) [BLS04] signature scheme from [FKL18].

[10] *Since no public-key encryption scheme can be secure information-theoretically.*

APPLICATIONS. Using the algebraic wrapper, we are able to transport several AGM-based proofs into the standard (group) model. For instance, the algebraic wrapper provides all necessary extraction properties for the tight security proof of Schnorr-signed ElGamal from [FPS20].

Thus, the algebraic wrapper provides protection of computations in the wrapped group in a relaxed manner which still suffices for several applications and can be instantiated from IO. We refer the reader to Part II of this thesis for more details.

1.3 PROTECTING COMPUTATIONS – USING COMPRESSION?

Lastly, we consider the classical notion of compression. Aside from the field of information theory [Sha48; Huf52; GM59; Flo64; Sch72; Ris76; Pas77; ZL77; ZL78; RJ79], starting from the seminal work of Goldberg and Sipser [GS85], compression has received considerable attention in the field of computational complexity theory [Wee04; TVZ05; HLR07].[11] Informally, given some distribution X, compression aims to efficiently encode samples from X as short strings while at the same time being able to efficiently recover these samples.

Prior to our work [ACI+20], it was entirely unclear how compression relates to the protection of computation. In this thesis and [ACI+20], we rigorously examine the notion and the properties of compression and vary this notion to escape classical impossibility results.[12] Subsequently, we shed light on an entirely unexpected connection between compression and the protection of computations in a distributed setting.

[11] *Complexity theory considers the case of compressing a single sample, whereas information theory clasically considers compression of multiple independent samples.*

[12] *If one-way functions exist, then there exist incompressible distributions (observation due to Levin).*

PROPERTIES OF COMPRESSION. Compression in its "perfect" form guarantees that compressed outputs are completely (information-theoretically) random leaving virtually no room for any further compression without losing information. From this angle, compression can be viewed as unifying two widely studied problems in computational complexity theory: *randomness condensers* [RR99; TV00; TUZ01; DRV12] and *resource-bounded Kolmogorov complexity* [Sol64; Kol68; Cha69; LV90; LV19]. Randomness condensers are functions that efficiently map an input distribution to an output distribution with higher entropy rate, i. e., with higher entropy per length ratio.[13] More precisely, a randomness condenser guarantees that its output entropy rate is higher than its input entropy rate. A randomness condenser can be viewed as an efficient compression algorithm – without a corresponding efficient decompression algorithm. On the other hand, the resource-bounded Kolmogorov complexity of a string is the smallest description length of an efficient program that outputs this string. For instance, the string "12121212121212" can be produced by a short program whereas the string "32576143764251" has no obvious simpler description than the string itself. Hence, the resource-bounded Kolmogorov complexity provides an intuitive measure of the amount of "randomness" in a particular string. Such a shortest program description can be viewed as a compressed string such that decoding is efficiently possible – while finding this compressed string may be inefficient.

[13] *Entropy measures the uncertainty or amount of randomness of a distribution.*

One central property of perfect compression – unifying the core properties of both randomness condensers and resource-bounded Kolmogorov complexity [GS85; TVZ05] – is the ability to efficiently map bitstrings from some distribution to uniformly random-looking bitstrings while at the same time allowing the original input to be efficiently recovered.

CRYPTOGRAPHIC APPLICATIONS OF COMPRESSION. To demonstrate the usefulness of this property, consider the following scenario. Suppose, Alice wants to confidentially communicate some (random) sequence of seven English words to Bob.[14] However, Alice and Bob only share a common password with *low entropy*, i. e., a password which can be efficiently guessed. Using the password as a secret key to encrypt Alice's message allows an eavesdropper to try all (most likely) passwords until decryption with a candidate password reveals a sequence of seven English words.

[14] *This message distribution is just a toy example. A more useful but more complex message distribution is the distribution modeling natural English language.*

This problem appears in the context of, e. g., password-authenticated key exchange [BM92], honey encryption [JR14] or subliminal communication and steganography [vHL05; CGOS07; HPRV19]. A natural solution is to use perfect compression as was done in [BMN01] to extend [BM92] for elliptic curve groups. Perfect compression maps samples from a distribution to random-looking strings. Compressing the message to a random-looking string before encrypting renders the above brute-force attack useless: decryption with a wrong password reveals a random-looking plaintext (by a standard assumption on the used secret-key encryption scheme) which is exactly the same behavior as decryption with the correct password.

However, perfect compression exists only for a very small number of message distributions. On the other hand, the core property of compression – that outputs are smaller than inputs – is not necessary for these applications. Indeed, efficiently encoding x into a (possibly longer) random-looking string suffices, provided that x can be efficiently recovered.

RELAXING COMPRESSION. Since compression (particularly perfect compression) only exists for a very small number of distributions, we vary the notion of compression as follows. We relax perfect compression by eliminating the requirement for shorter outputs and analyze several variants of the resulting notion, see also Figure 1.4 for an overview. Moreover, we consider variants assuming the presence of a trusted setup in the form of a common reference string.

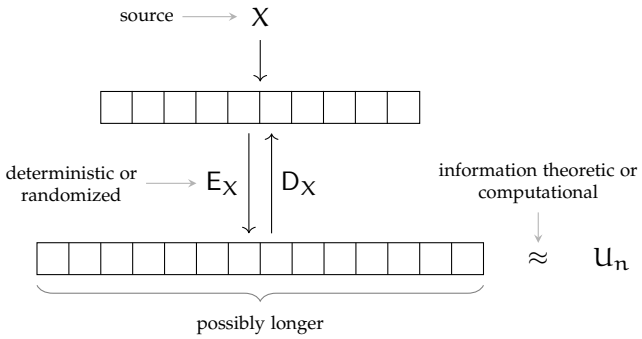


Figure 1.4: Illustration of our notion of pseudorandom encodings. Compared to perfect compression, we drop the requirement for shorter outputs, we consider deterministic and randomized encoding algorithms and we consider computational and information-theoretic indistinguishability from randomness. Note that we also study variants assuming a trusted setup which are are not captured by this visual.

The resulting notion only preserves the properties that encoded samples can be efficiently recovered and that encoded samples appear to be random. For this reason, we refer to this notion as *pseudorandom encodings*. A pseudorandom encoding scheme for a distribution X is a tuple (E_X, D_X), where E_X is an efficient (possibly randomized) encoding algorithm and D_X is an efficient deterministic decoding algorithm satisfying the following two properties.

CORRECTNESS. Decoding works with overwhelming probability. More formally,

$$\Pr\left[y \leftarrow X_\lambda : D_X(E_X(y)) = y\right]$$

is overwhelming.

PSEUDORANDOMNESS. An encoded sample is indistinguishable from true randomness. More formally,

$$E_X(X_\lambda) \approx U_{n(\lambda)},$$

where $n(\lambda)$ is the output length of E_X and "$\approx$" denotes some notion of indistinguishability.

Additionally, we consider distributions X which can accept an input $m \in L \subseteq \{0,1\}^*$, i. e., distribution families $(X_m)_{m \in L}$. In this case, the above properties need to hold for all such inputs.

Our main focus is on the hypothesis that *all* efficiently samplable distributions can be pseudorandomly encoded. We refer to this hypothesis as the *pseudorandom encoding hypothesis* (PREH).

The above notion offers several degrees of freedom: the encoding algorithm can be deterministic or randomized and the indistinguishability requirement can be information-theoretical or computational. Additionally, we may assume a trusted setup, i. e., a globally known common reference string (CRS) which is set up honestly. If a distribution X admits an input $m \in L$, this further entails two different notions: a non-adaptive version, where correctness and pseudorandomness hold only for inputs m which are independent of the CRS, and an adaptive version, where correctness and pseudorandomness hold also for inputs m which are chosen depending on the CRS. See Figure 1.5 for an overview of these notions.

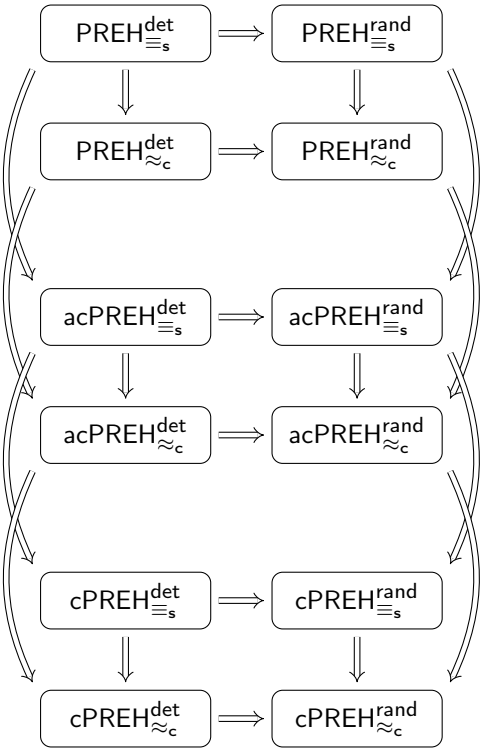


Figure 1.5: An overview of the different notions of pseudorandom encodings. The superscripts "det" and "rand" indicate whether the encoding algorithm is deterministic or allowed to be randomized. The subscripts "$\equiv_s$" and "$\approx_c$" indicate whether the guarantees are information-theoretical or computational. The prefixes "c" and "ac" indicate that a trusted setup is assumed where inputs are required to be chosen statically or adaptively, respectively.

In Part III of this thesis which is based on the publication [ACI+20], we provide a classification of pseudorandom encodings and identify computational, randomized pseudorandom encodings with common setup as the best possible and useful notion, and instantiate this building block from IO.

In other words, assuming that we can protect the computations inside some program from the user of the program (i. e., if IO exists), then pseudorandom encoding schemes exist for all efficiently samplable distributions in their weakest variant. However, this is not the only common ground between pseudorandom encodings and the protection of computations as we will elaborate in the following.

PROTECTING COMPUTATIONS. Surprisingly, the resulting novel notion of pseudorandom encodings turns out to be strongly connected to *fully adaptively* secure multi-party computation.[15] Multi-party computation (MPC) allows parties to jointly compute any possibly randomized program on their inputs without revealing them. More precisely, while every party eventually learns the output, nothing beyond what follows from that party's input and output can be inferred. In particular, no inputs from other parties can be learned.

[15] *We consider the feasibility of MPC for* all *(possibly randomized) programs.*

Security in such a setting is defined by comparing the actual protocol execution (the real world) with an ideally secure protocol in the presence of an adversary who may take control of parties. If every adversarial behavior in the real world can be emulated in the ideally secure world, then the real world protocol is "at least as secure as" the ideally secure protocol [GMW87]. The greatest challenge pose *adaptive* adversaries who may adaptively decide to corrupt any number of participating parties at any time during the protocol execution. This behavior is particularly challenging to emulate.

In other words, multi-party computation allows for executing *protected computations* in settings, where many mutually distrustful parties are involved. This manner of protecting computations should be contrasted with the way code obfuscation protects computations. Code obfuscation conceals the way *how* a program derives outputs from entities who evaluate the program. In the setting of multi-party computation, on the other hand, every party may very well know the program which is to be jointly evaluated. However, multi-party computation guarantees that no party is able to infer a single intermediate value which does not already follow from that party's input and output.

The notion of adaptively secure MPC was first considered in [BH93; CFGN96]. [BH93] relies on secure and trusted erasures. However, since erasing data is not verifiable and may be subject to hardware limitations, it is (arguably) not satisfying to rely on other parties to erase their internal data. Hence, we only consider a setting, where secure erasures are not assumed. There are several positive results on the feasibility of adaptive MPC in restricted settings such as the honest majority setting (where a majority of the parties remains uncorrupted) [CFGN96; Bea97], the all-but-one corruptions setting (where at least one party remains uncorrupted) [KO04; IPS08; GS12; HP14; DPR16], or restricting the class of supported functions to so-called "well-formed" ones (which do not hide their internal randomness) [CLOS02]. We consider the strongest notion of adaptive security for *all* randomized functionalities allowing adaptive corruptions of *all* parties [IKOS10; DKR15; CPV17]. A setting, where all parties are corrupted may seem unnatural. However, as outlined in [DKR15], a protocol running a sub-protocol among a strict subset of the parties should remain secure even if all parties running the sub-protocol are corrupted.

HOW TO EMULATE ADAPTIVE CORRUPTION OF ALL PARTIES. Consider the adversary who merely observes the entire protocol execution and decides to corrupt all parties after the protocol has terminated. Since secure erasures are not assumed, this adversary learns well-distributed random tapes which explain all messages sent by the individual parties. Recovering such valid-looking random tapes *after the output is fixed* is a very challenging task.

Naively speaking, invertible sampling constitutes the problem to invert randomized algorithms, i. e., to produce a valid-looking random tape that leads to a given output. However, there exist several randomized algorithms which cannot be inverse sampled to this extent. Consider, for instance, the algorithm which randomly samples two large prime numbers and outputs their product. Given our current understanding of computational number theory, this algorithm cannot be inverse sampled in the above sense. In fact, invertible sampling due to [CFGN96; DN00; GKM+00; IKOS10] only requires the existence of an inverse samplable algorithm which induces an output distribution similar to the output distribution of the actual sampler. This notion of invertible sampling is exactly what is both necessary and sufficient to emulate fully adaptive adversarial behavior. Ishai, Kumarasubramanian, Orlandi, and Sahai [IKOS10] establish an equivalence between invertible sampling and *fully adaptively secure* multi-party computation (MPC). More precisely, fully adaptively secure multi-party computation is possible if and only if invertible sampling is possible for all efficiently samplable distributions with input.

Our study of pseudorandom encodings reveals a surprising connection between invertible sampling and pseudorandom encodings: our novel notion of pseudorandom encodings emerges to be equivalent to invertible sampling. Consequently, pseudorandom encodings are equivalent to fully adaptively secure MPC – the strongest form of protecting computations in a distributed setting.

Our study of pseudorandom encodings unveils further unexpected connections between several fields within cryptography, like fully adaptive MPC, honey encryption and steganography. We refer the reader to Part III of this thesis for more details.

SUBSEQUENT WORK. In [ACI+20], we obtain partial evidence that indistinguishability obfuscation is not only sufficient for pseudorandom encodings but also *necessary*. Hence, we raised the question whether IO is indeed necessary for pseudorandom encodings. This open question transpired quite helpful for the development of plausible IO candidates. In fact, shortly after uploading a preliminary version of [ACI+20] to the IACR cryptology ePrint archive, Wee and Wichs [WW20] answer our question positively giving rise to a novel construction of IO from the assumption that pseudorandom encoding schemes exist for a very small class of distributions (and the learning with errors assumption).

1.4 STRUCTURE OF THIS THESIS

This thesis is divided into 3 parts. Each part contains an introduction including a short summary of the contribution presented in that part and a discussion setting the results in context to the state-of-the-art. These introductions are taken verbatim from the works [ACH20; AHK20; ACI+20] with minor changes.

Before we present our contributions in Parts I, II and III in detail, Chapter 2 introduces basic notation and definitions of cryptographic primitives which we will use in this thesis. Chapter 2 is in part taken verbatim from the works [ACH20; AHK20; ACI+20].

- In Part I of this thesis, we present our results regarding probabilistic indistinguishability obfuscation. We introduce a novel framework called "doubly-probabilistic indistinguishability obfuscation" which allows to avoid subexponentially secure IO in several contexts. Part I is taken verbatim from the publication [ACH20] with minor changes.
- In Part II of this thesis, we present our results on instantiating the algebraic group model from falsifiable assumptions. We introduce a novel framework called "algebraic wrapper" which can be instantiated in the standard model and allows to transport several proofs from the algebraic group model into the standard model. Part II is taken verbatim from the publication [AHK20] with minor changes.
- In Part III of this thesis, we introduce our results on pseudorandom encodings. We introduce the novel notion of pseudorandom encodings and provide a unified study including positive and negative results on the feasibility of several flavors of pseudorandom encodings. Furthermore, we demonstrate how pseudorandom encodings relate to various areas in cryptography and derive new insights for those fields. Part III is taken verbatim from the publication [ACI+20] with minor changes.

Finally, in Chapter 19, we provide a brief outlook on possible further research directions.

2 PRELIMINARIES

In this chapter, we introduce notations and technical preliminaries which are common to the subsequent Parts I, II and III. The preliminaries are in part taken verbatim from the works [ACH20; AHK20; ACI+20].

2.1 NOTATIONS

For $n \in \mathbb{N}$, we denote by $[n]$ the set $\{1, \ldots, n\}$. We denote the set of all functions mapping from set $\mathcal{X}$ to $\mathcal{Y}$ by $\mathit{maps}(\mathcal{X}, \mathcal{Y})$. The security parameter is denoted by λ and is implicitly given to all algorithms in unary. A probabilistic polynomial time (PPT) algorithm runs in time polynomial in the (implicit) security parameter λ. We consider non-uniform polynomial time adversaries, i. e., polynomial time adversaries receiving a polynomially bounded auxiliary input (also called "advice string") depending only on the security parameter.[16] A function $\mathit{negl}\colon \mathbb{N} \to \mathbb{R}$ is negligible in λ if for every constant $c \in \mathbb{N}$, there exists a bound $n_c \in \mathbb{N}$, such that for all $n \geqslant n_c$, $|\mathit{negl}(n)| \leqslant n^{-c}$. In asymptotic notation, the set of negligible functions is $\lambda^{-\omega(1)}$. A function g is overwhelming if $1 - g(\lambda)$ is a negligible function.

Given a finite set $\mathcal{X}$, the notation $x \leftarrow \mathcal{X}$ means a uniformly random assignment of an element of $\mathcal{X}$ to the variable x. Given a probability distribution D, the notation $x \leftarrow D$ means sampling an element according to the distribution D and assigning that element to x. The support of a probability distribution D is the set of all elements with a non-zero probability to occur and is denoted by $\mathit{supp}(D)$. We denote the uniform distribution over bitstrings of length n by U_n. We use the notation $\{C_1, C_2, \ldots, C_m \colon D\}$ to denote the distribution of D which is obtained as the result of the process defined by the sequence of instructions $C_1, \ldots, C_m$.

Let E_1 and E_2 be events of some probabilistic process. We denote the probability that E_1 occurs by $\Pr[E_1]$, and the probability that E_1 occurs conditioned on the fact that the event E_2 occurred by $\Pr[E_1 \mid E_2]$.

Given a possibly randomized algorithm A, the notation $y \leftarrow A(x)$ means evaluation of A on input of x with fresh random coins r and assignment to the variable y. Sometimes we make the random coins r used by A explicit and write $A(x; r)$. The notation $\mathcal{A}^{\mathcal{O}}$ indicates that the algorithm $\mathcal{A}$ is given oracle access to $\mathcal{O}$.

When describing security games, we always consider *stateful* adversaries unless explicitly stated otherwise. In some cases, we emphasize this by handling the adversarial state (usually denoted by *st*) explicitly.

In game based proofs, out_i denotes the output of game $\mathbf{G}_i$. Further, we will use this notation to highlight differences to previous hybrids.

[16] *Note that by a coin-fixing argument, it is sufficient to consider non-uniform deterministic adversaries. Most of the results in this thesis apply for uniform PPT adversaries as well. In case we make explicit use of the non-uniformity of the adversary, we remark this explicitly.*

DEFINITION 2.1 (Statistical distance). The *statistical distance* Δ between two probability distributions X and Y (over some set $\mathcal{X}$) is

$$\Delta(X, Y) \coloneqq \sum_{x \in \mathcal{X}} \big|\Pr[X = x] - \Pr[Y = x]\big|.$$

In other words, the statistical distance is the metric induced by the Manhattan norm (or ℓ_1 norm) on the probability space. We say that two distributions are statistically close if their statistical distance is negligible.

DEFINITION 2.2 (Min-entropy). The *min-entropy* of a distribution X is

$$H_\infty(X) = -\log\Big(\max_{x \in supp(X)} \Pr[X = x]\Big).$$

In other words, the probability of correctly guessing the outcome of a probability experiment with min-entropy μ is at most $2^{-\mu}$.

In many cases, some information Z that is correlated to the actual source X is known. Since for our purposes, the conditional part Z is not under adversarial control, we use the notion of *average* conditional min-entropy as used in [HLR07; DORS08].

DEFINITION 2.3 (Average min-entropy, [HLR07; DORS08]). Let (X, Z) be a joint distribution. The *average min-entropy* of X conditioned on Z is

$$\widetilde{H}_\infty(X \mid Z) \coloneqq -\log\Big(\mathop{\mathbb{E}}_{z \leftarrow Z}\big[\max_x \Pr[X_z = x]\big]\Big).$$

Extractors allow to obtain (almost) uniform randomness from samples from any weakly random source of entropy, as long as a short uniform random seed is available. Hence, even weakly random distributions can be useful for cryptography.

DEFINITION 2.4 (Extractor, [ILL89]). A function $\mathsf{Ext}\colon \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ is a (k, ϵ)-*extractor* if for all distributions X on $\{0,1\}^n$ with min-entropy $H_\infty(X) \geqslant k$ (which are independent of the key K), the statistical distance

$$\Delta\Big(\{K \leftarrow U_d\colon (K, \mathsf{Ext}(K, X))\}, \{K \leftarrow U_d\colon (K, U_m)\}\Big)$$

is at most ϵ.

Initially, such extractors were used to reduce the randomness complexity of algorithms, [Sip88; San87; NZ96].

2.2 HARDNESS ASSUMPTIONS

Cryptography aims to provide an asymmetry between the computational cost for an honest user and the computational cost for adversaries. The honest user is provided with computationally affordable mechanisms to, e. g., protect information, whereas it should be computationally costly for an adversary to obtain this protected information. Asymptotic security is the complexity theoretic approach for achieving this. The computational costs for user and adversary are parametrized by a *security parameter* $\lambda \in \mathbb{N}$. Tasks that are computable in probabilistic polynomial-time in the security parameter are considered efficient, all other tasks are considered infeasible

(or intractable). The philosophy behind asymptotic security is that one is able to scale the security guarantee (by choosing a suitable value for λ) thus covering arbitrarily powerful adversaries. A security proof consists of a reduction[17] of a computational problem to the task of breaking the security guarantee in question.

The validity of solutions for adversarial tasks in cryptography can be efficiently tested. In other words, the adversarial tasks of interest to cryptography are in $\mathcal{NP}$.[18] Thus, a necessary condition for the existence of (most) cryptographic schemes is $\mathcal{BPP} \not\subseteq \mathcal{NP}$ (implying $\mathcal{P} \neq \mathcal{NP}$).

The complexity classes $\mathcal{P}$ and $\mathcal{NP}$ are only concerned with the worst-case complexity. However, worst-case hardness does not suffice for cryptography since finding hard instances must be tractable for the honest users (i. e., must be computable in probabilistic polynomial-time). Therefore, for cryptography, we assume the *average-case* hardness of problems in $\mathcal{NP}$.

The fundamental notion of average-case hardness was conceived by Levin [Lev86]. Average-case hardness against polynomial-time adversaries for computational and decisional problems is roughly defined as follows.

DEFINITION 2.5 (Distributional search problem (informal), [Lev86; IL90]). A *distributional search problem* is a tuple (S, μ), where $S \subseteq \{0,1\}^* \times \{0,1\}^*$ and $\mu\colon \{0,1\}^* \to [0,1]$ is a distribution over inputs.

DEFINITION 2.6 (Distributional decision problem (informal), [Lev86; IL90]). A *distributional decision problem* is a tuple (D, μ), where $D\colon \{0,1\}^* \to \{0,1\}$ and $\mu\colon \{0,1\}^* \to [0,1]$ is a distribution over inputs.

A decision problem (D, μ) is in distributional $\mathcal{NP}$ if μ is computable in polynomial time (that is efficiently samplable) and D is an $\mathcal{NP}$-predicate. Furthermore, the class of distributional $\mathcal{NP}$ contains complete problems.

For cryptography, a more direct way of formalizing average-case hardness emerged. Instead of considering one fixed input distribution, in cryptography, we consider a sequence of input distributions $(\mu_\lambda)_{\lambda\in\mathbb{N}}$ such that μ_λ is a distribution over $\{0,1\}^\lambda$.

DEFINITION 2.7 (Average-case hardness (informal), [Lev86; GM82; BM82; Yao82]). A distributional decision problem $(D, (\mu_\lambda)_{\lambda\in\mathbb{N}})$ (resp., search problem $(S, (\mu_\lambda)_{\lambda\in\mathbb{N}})$) is *hard on the average* if $(\mu_\lambda)_{\lambda\in\mathbb{N}}$ is efficiently samplable (in λ) and for all PPT adversaries, the probability to correctly solve D is at most negligibly better than guessing (resp., at most negligible), where the probability is also over the input choice from $(\mu_\lambda)_{\lambda\in\mathbb{N}}$.

SUBEXPONENTIAL HARDNESS. There are two common ways to define hardness against subexponential adversaries. The weaker variant requires the advantage of polynomial-time adversaries to decrease subexponentially in the security parameter, i. e., requires the advantage to be in $2^{-o(\lambda)}$.[19] This definition is commonly used in the context of subexponentially secure IO. The stronger variant requires the advantage of subexponential-time adversaries to decrease subexponentially in the security parameter.

NON-UNIFORM ADVERSARIES. A stronger model of computation considers so-called non-uniform adversaries. This model of computation originates from the complexity class $\mathcal{P}/\mathrm{poly}$ containing all languages which are decidable by polynomial-sized circuits. A non-uniform (PPT) adversary is a tuple

[17] *A reduction from a problem A to a problem B is a probabilistic polynomial-time algorithm which solves problem A (with non-negligible probability) given oracle access to a problem solver for B.*

[18] *For simplicity, we do not make a distinction between decision and search problems at this point.*

[19] *That is, the advantage is at most $2^{-\lambda^\epsilon}$ for some $0 \leqslant \epsilon < 1$.*

$(\mathcal{A}, (a_\lambda)_{\lambda\in\mathbb{N}})$, where $\mathcal{A}$ is a (probabilistic) polynomial-time machine and $(a_\lambda)_{\lambda\in\mathbb{N}}$ is a sequence of polynomial-sized bitstrings. That is, a_λ provides $\mathcal{A}$ with one additional advice string per input length. Note that by a coin-fixing argument, non-uniform PPT adversaries and non-uniform deterministic polynomial-time adversaries are equivalent.

2.2.1 Efficient Falsifiability

An assumption is falsifiable if an observation can show it to be false. Being able to assess the plausibility of hardness assumptions is an important task. For this purpose, Naor [Nao03] introduces a classification of hardness assumptions based on the *complexity of falsifying* them. Informally, a cryptographic assumption is falsifiable if the assumption can be formalized as an interaction between an *efficient* challenger and an adversary, such that the challenger can efficiently provide the adversary with a challenge and is able to efficiently verify the adversary's solution attempt. Naor classifies assumptions into four classes. These are "efficiently falsifiable" assumptions, "falsifiable" assumptions, "somewhat falsifiable" assumptions[20] and "non-falsifiable assumptions" (ordered decreasing in plausibility). Assumptions which are at least somewhat falsifiable are acceptable for use in cryptography. This, however, is (arguably) not true for non-falsifiable assumptions.[21]

Note that falsifiability is not closed under security reductions. There are security properties that are non-falsifiable (when formulated as an assumption) but can be instantiated from falsifiable assumptions. For instance, the soundness property of non-interactive zero-knowledge (NIZK) proof systems (see Section 2.8) is one such property. In this thesis, we only use falsifiable assumptions and security properties which can be based on falsifiable assumptions.

[20] *The difference between efficient falsifiability, falsifiability and somewhat falsifiability is mostly the runtime to verify the solution.*

[21] *Using non-falsifiable assumptions might be better than simply assuming "the scheme is secure", but a security reduction based on such assumptions is (arguably) not satisfying.*

2.3 ONE-WAY FUNCTIONS

Before we define one-way functions, we define the notion of efficiently computable function family ensembles.

DEFINITION 2.8 (Function family ensemble). Let $n(\lambda), m(\lambda)$ be polynomially bounded length functions. A *function family ensemble* is a tuple of PPT algorithms $\mathcal{F} = (\mathsf{Gen}, \mathsf{Eval})$ such that

$$\left\{\mathsf{Eval}(f,\cdot)\colon \{0,1\}^{n(\lambda)} \to \{0,1\}^{m(\lambda)} \;\middle|\; f \in \mathit{supp}(\mathsf{Gen}(1^\lambda))\right\}_{\lambda\in\mathbb{N}}$$

is an efficiently computable family of functions.

For ease of notation, we often write $f(x)$ instead of $\mathcal{F}.\mathsf{Eval}(f,x)$. A one-way function is a function on bitstrings which can be efficiently evaluated but which is hard to invert given the image of a random input.

DEFINITION 2.9 (One-way function, [DH76]). A *one-way function (OWF)* family ensemble is a function family ensemble $\mathcal{F} = (\mathsf{Gen}, \mathsf{Eval})$, such that for all PPT adversaries $\mathcal{A}$,

$$\mathrm{Adv}^{\mathsf{ow}}_{\mathcal{F},\mathcal{A}}(\lambda) \coloneqq \Pr\left[\begin{array}{lll} f & \leftarrow \mathsf{Gen}(1^\lambda) & \\ x & \leftarrow U_{n(\lambda)} & : f(x') = f(x) \\ x' & \leftarrow \mathcal{A}(f, f(x)) & \end{array}\right]$$

is negligible in λ.

The existence of one-way functions is the most fundamental hardness assumption in cryptography.

2.4 PUNCTURABLE PSEUDORANDOM FUNCTIONS

Pseudorandom functions are functions on bitstrings which are computationally indistinguishable from truly random functions as long as the function key remains secret.

DEFINITION 2.10 (Pseudorandom function, [GGM86]). A family of *pseudorandom functions (PRFs)* is a tuple of PPT algorithms $\mathsf{F} = (\mathsf{KGen}, \mathsf{Eval})$ and two computable functions $n(\lambda)$, $m(\lambda)$ such that

- $\mathsf{KGen}(1^\lambda)$ outputs a key K, and
- $\mathsf{Eval}(K, \cdot)$, given a key $K \in \mathit{supp}(\mathsf{KGen}(1^\lambda))$, defines a function mapping bitstrings of length $n(\lambda)$ to bitstrings of length $m(\lambda)$.

A PRF F is required to satisfy the following property.

$$\mathrm{Adv}^{\mathsf{prf}}_{\mathsf{F},\mathcal{A}}(\lambda) \coloneqq \Big|\Pr\left[K \leftarrow \mathsf{KGen}(1^\lambda) : \mathcal{A}^{\mathsf{F.Eval}(K,\cdot)}(1^\lambda) = 1\right] - \Pr\left[R \leftarrow \mathit{maps}(\{0,1\}^{n(\lambda)}, \{0,1\}^{m(\lambda)}) : \mathcal{A}^{R(\cdot)}(1^\lambda) = 1\right]\Big|$$

is a negligible function in λ.

For ease of notation we often write $\mathsf{F}(K, x)$ to denote $\mathsf{F.Eval}(K, x)$. Pseudorandom functions can be constructed from one-way functions [GGM86].

For some applications, it will be useful to generate PRF keys which can be used to evaluate the PRF everywhere except for a pre-defined subset of preimages. This procedure is called "puncturing". A punctured PRF key is not only insufficient to evaluate the PRF at a punctured point, but it also does not leak any information about the actual PRF images at punctured points.

DEFINITION 2.11 (Puncturable PRF, [GGM86; BW13; BGI14; KPTZ13]). A family of *puncturable PRFs (pPRFs)* is a tuple of PPT algorithms $\mathsf{F} = (\mathsf{KGen}, \mathsf{Punct}, \mathsf{Eval})$ and two computable functions $n(\lambda)$, $m(\lambda)$ such that

- $\mathsf{KGen}(1^\lambda)$ outputs a key K,
- $\mathsf{Punct}(K, T)$, on input of a key K, and a subset of the input space $T \subset \{0,1\}^{n(\lambda)}$, outputs a punctured key $K\{T\}$,
- $\mathsf{Eval}(K', \cdot)$, on input of a key K' defines a function mapping bitstrings of length $n(\lambda)$ to bitstrings of length $m(\lambda)$. $\mathsf{Eval}(K\{T\}, x)$ on input of a punctured key $K\{T\}$ and an $x \in T$, outputs a special symbol $\perp$.

A pPRF is required to satisfy the following properties.

FUNCTIONALITY PRESERVED UNDER PUNCTURING. For all PPT adversaries $\mathcal{A}$ which output a set $T \subseteq \{0,1\}^{n(\lambda)}$, we have that

$$\Pr\left[\begin{array}{l} T \leftarrow \mathcal{A}(1^\lambda) \\ K \leftarrow \mathsf{KGen}(1^\lambda) \\ K\{T\} \leftarrow \mathsf{Punct}(K,T) \end{array} : \begin{array}{c} \forall x \in \{0,1\}^{n(\lambda)} \setminus T: \\ \mathsf{Eval}(K,x) = \mathsf{Eval}(K\{T\},x) \end{array}\right] = 1.$$

PSEUDORANDOM AT PUNCTURED POINTS. For all PPT adversaries $\mathcal{A}$,

$$\mathrm{Adv}^{\text{s-pprf}}_{F,\mathcal{A}}(\lambda) := \left|\Pr\left[\mathrm{Exp}^{\text{s-pprf}}_{F,\mathcal{A},0}(\lambda) = 1\right] - \Pr\left[\mathrm{Exp}^{\text{s-pprf}}_{F,\mathcal{A},1}(\lambda) = 1\right]\right|$$

is negligible, where $\mathrm{Exp}^{\text{s-pprf}}_{F,\mathcal{A},b}(\lambda)$ is defined in Figure 2.1.

$\mathrm{Exp}^{\text{s-pprf}}_{F,\mathcal{A},b}(\lambda)$

$(T, st) \leftarrow \mathcal{A}(1^\lambda)$
$K \leftarrow \mathsf{F.KGen}(1^\lambda)$
$K\{T\} \leftarrow \mathsf{F.Punct}(K,T)$
$y_0 \leftarrow \mathsf{F.Eval}(K,T)$
$y_1 \leftarrow \{0,1\}^{m(\lambda)}$
return $\mathcal{A}(st, K\{T\}, y_b)$

Figure 2.1: Selective security game $\mathrm{Exp}^{\text{s-pprf}}_{F,\mathcal{A},b}(\lambda)$ for puncturable pseudorandom functions.

[BW13; BGI14; KPTZ13] observed that the original tree-based PRF construction from [GGM86] already satisfies this definition. In this thesis, we will additionally be concerned with a special type of pPRFs which allow to extract (almost) uniformly random strings from high-entropy inputs, even when the function key is public (or not guaranteed to be hidden, as is the case in many applications involving indistinguishability obfuscation). That is, pPRFs which also act as randomness extractors (cf. Definition 2.4).

DEFINITION 2.12 (Extracting pPRF, [SW14]). An *extracting* puncturable PRF family with error $\epsilon(\cdot)$ for min-entropy $k(\cdot)$ is a family of puncturable PRFs F mapping $n(\lambda)$ bits to $m(\lambda)$ bits such that for all $\lambda \in \mathbb{N}$, if X is a distribution over $\{0,1\}^{n(\lambda)}$ with min-entropy at least $k(\lambda)$, then the statistical distance

$$\Delta\Big(\{K \leftarrow \mathsf{KGen}(1^\lambda): (K, F(K,X))\}, \{K \leftarrow \mathsf{KGen}(1^\lambda): (K, U_{m(\lambda)})\}\Big)$$

is at most $\epsilon(\lambda)$.

Due to [SW14] (full version, Theorem 3), extracting pPRFs exist assuming one-way functions. On a high level, Sahai and Waters [SW14] show that composing a puncturable PRF (which is almost always injective) with a strong extractor yields an extracting pPRF.

2.5 OBFUSCATION

Code obfuscation has been formally introduced in [Hado0; BGI+01; BGI+12]. An obfuscator is an efficient compiler which transforms a given program into an unintelligible one while preserving its original functionality. Several notions of "unintelligibility" for program obfuscation have been studied. The strongest conceivable notion is virtual black-box (VBB) obfuscation. This notion requires that the obfuscated program is a "virtual black-box", i. e., does not leak anything about the original program except for its input-output behavior. More formally, everything being efficiently computable given the obfuscated program, can also be efficiently computed given mere oracle access to the original program. However, Barak et al. [BGI+01; BGI+12] show that if one-way functions exist, there are inherently unobfuscatable programs. On the other hand, the existence of a VBB obfuscator (for a relatively small class of programs) implies the existence of one-way functions. Hence, VBB obfuscation is unconditionally impossible.

Barak et al. [BGI+01; BGI+12] also introduce a weaker notion of obfuscation called "indistinguishability obfuscation" (IO). An indistinguishability obfuscator only guarantees that if two programs are functionally equivalent, then their obfuscations are computationally indistinguishable. This definition avoids the black-box paradigm along with the impossibility results for VBB obfuscation. However, intuitively, this notion is rather limited compared to VBB since there are no guarantees for circuits which compute different functions. Barak et al. [BGI+01; BGI+12] show that an inefficient indistinguishability obfuscator exists unconditionally. Namely, the obfuscator which on input of a Boolean circuit C outputs the lexicographically first circuit which computes the same function as C and has size $|C|$ is an inefficient indistinguishability obfuscator. If $\mathcal{P} = \mathcal{NP}$, the polynomial hierarchy collapses and finding such an obfuscation is efficiently possible. Hence, the existence of an indistinguishability obfuscator cannot imply one-way functions.

Later, Goldwasser and Rothblum [GR07] show that indistinguishability obfuscation is equivalent to "best-possible" obfuscation. An obfuscator is a best-possible obfuscator if any information that is not hidden by the obfuscated circuit, is also not hidden by any other circuit of similar size which computes the same function. This is formalized via a simulator. Namely, a best-possible obfuscator $\mathcal{O}$ guarantees that for every adversary (or "learner") $\mathcal{L}$, there exists a simulator Sim such that for any pair of functionally equivalent circuits C_1, C_2 (of identical length), the distributions $\mathcal{L}(\mathcal{O}(C_1))$ and $\mathsf{Sim}(C_2)$ are computationally indistinguishable. That is, no obfuscator can hide more information than a best-possible obfuscator. Further, Goldwasser and Rothblum [GR07] show that indistinguishability obfuscation is actually equivalent to best-possible obfuscation.

INDISTINGUISHABILITY OBFUSCATION. Let $\mathcal{C} = (\mathcal{C}_\lambda)_{\lambda\in\mathbb{N}}$ be an ensemble of sets $\mathcal{C}_\lambda$ of Boolean circuits. The set $\mathcal{C}_\lambda$ contains circuits of size λ with input length $n(\lambda)$.

DEFINITION 2.13 (Indistinguishability obfuscation, [GGH+13]). A uniform PPT algorithm iO is an *indistinguishability obfuscator (IO)* for a circuit class $\mathcal{C} = (\mathcal{C}_\lambda)_{\lambda\in\mathbb{N}}$ if the following conditions are satisfied.

CORRECTNESS. For all security parameters $\lambda \in \mathbb{N}$, all circuits $C \in \mathcal{C}_\lambda$, all inputs x, we have that

$$\Pr\left[\Lambda \leftarrow \mathsf{iO}(C) \colon \Lambda(x) = C(x)\right] = 1.$$

SECURITY. For every (possibly non-uniform) PPT distinguisher $\mathcal{A}$, for all pairs of circuits $(C_0, C_1) \in \mathcal{C}_\lambda \times \mathcal{C}_\lambda$ such that for all inputs x, $C_0(x) = C_1(x)$, we have that

$$\mathrm{Adv}^{\text{io-ind}}_{\mathsf{iO},\mathcal{A}}(\lambda) \coloneqq \left|\Pr\left[\mathcal{A}(1^\lambda, \mathsf{iO}(C_0)) = 1\right] - \Pr\left[\mathcal{A}(1^\lambda, \mathsf{iO}(C_1)) = 1\right]\right|$$

is negligible in λ.

Definition 2.13 can also be formulated with respect to an efficient probabilistic circuit sampler D for $\mathcal{C}$. Intuitively, this corresponds to a "legitimate" adversary choosing the circuit pair (C_0, C_1) in a "first phase".

A circuit sampler for $\mathcal{C}$ is defined as a family of efficiently samplable distributions $D = (D_\lambda)_{\lambda\in\mathbb{N}}$, where D_λ is a distribution over triplets (C_0, C_1, z) with $C_0, C_1 \in \mathcal{C}_\lambda$ such that C_0 and C_1 take inputs of the same length and $z \in \{0,1\}^{poly(\lambda)}$ is some state information. This state information z allows to pass information which the circuit sampler D generates alongside (C_0, C_1) to the adversary $\mathcal{A}$. We write $D(1^\lambda)$ to denote the efficient sampling process according to the distribution D_λ.

DEFINITION 2.14 (Functionally equivalent circuit sampler). We call a circuit sampler $D = (D_\lambda)_{\lambda\in\mathbb{N}}$ a *functionally equivalent circuit sampler* if D guarantees that the circuits C_0 and C_1 are functionally equivalent in the sense that for all inputs x, we have $C_0(x) = C_1(x)$.

The alternative definition of indistinguishability obfuscation relative to functionally equivalent circuit samplers is as follows.

DEFINITION 2.15 (Indistinguishability obfuscation (alternative), [GGH+13]). A uniform PPT algorithm iO is an *indistinguishability obfuscator (IO)* for a circuit class $\mathcal{C} = (\mathcal{C}_\lambda)_{\lambda\in\mathbb{N}}$ if the following conditions are satisfied.

CORRECTNESS. For all security parameters $\lambda \in \mathbb{N}$, all circuits $C \in \mathcal{C}_\lambda$, all inputs x, we have that

$$\Pr\left[\Lambda \leftarrow \mathsf{iO}(C) \colon \Lambda(x) = C(x)\right] = 1.$$

SECURITY. For every (possibly non-uniform) PPT distinguisher $(D, \mathcal{A})$, where D is a functionally equivalent circuit sampler for $\mathcal{C}$, we have that

$$\begin{aligned}\mathrm{Adv}^{\text{io-ind}}_{\mathsf{iO},D,\mathcal{A}}(\lambda) \coloneqq \\ \Big|\Pr\left[(C_0, C_1, z) \leftarrow D_\lambda \colon \mathcal{A}(1^\lambda, C_0, C_1, z, \mathsf{iO}(C_0)) = 1\right] \\ - \Pr\left[(C_0, C_1, z) \leftarrow D_\lambda \colon \mathcal{A}(1^\lambda, C_0, C_1, z, \mathsf{iO}(C_1)) = 1\right]\Big|\end{aligned}$$

is negligible in λ.

For non-uniform adversaries, an indistinguishability obfuscator according to Definition 2.13 for a circuit class $\mathcal{C}$ is also an indistinguishability obfuscator according to Definition 2.15 for the circuit class $\mathcal{C}$. This can be realized by a coin-fixing argument. We can fix a worst-case choice of random

coins r for D_λ. Given random coins r, D_λ produces concrete (C_0, C_1, z) for which $\mathcal{A}$'s advantage is still negligible due to Definition 2.13. For this thesis, Definition 2.15 will be sufficient.

[JLS20] provides the first instantiation of IO based on well-founded falsifiable assumptions.

IO IS EFFICIENTLY FALSIFIABLE. Definition 2.15 enjoys efficient falsifiability due to Naor [Nao03] as opposed to Definition 2.13. The existence of an indistinguishability obfuscator according to the original definition from [BGI+12] (see Definition 2.13) falls within the category of non-falsifiable assumptions. Definition 2.13 can be modeled as a game between an adversary and a challenger. The adversary sends a pair of circuits (C_0, C_1) to the challenger who obfuscates a random one of them and returns this obfuscation to the adversary who then has to guess which one was obfuscated. However, the challenger is unable to efficiently test whether the adversarially chosen circuits C_0 and C_1 are functionally equivalent, i. e., if for all inputs x, $C_0(x) = C_1(x)$. This would require the challenger to solve the SAT instance $C_0(x) \oplus C_1(x)$.

The alternative formulation from Definition 2.15 requires that no adversary can distinguish obfuscations of a circuit pair (C_0, C_1) which is (honestly) sampled from a functionally equivalent circuit sampler (cf. Definition 2.14). Hence, Definition 2.15 does satisfy efficient falsifiability.

DIFFERING-INPUTS OBFUSCATION. The notion of differing-inputs obfuscation (dIO) was defined in [BGI+12; BCP14; ABG+13] as a generalization of IO. The notion of dIO is defined very similarly to Definition 2.15 except that security is required to hold not only for functionally equivalent circuit samplers (see Definition 2.14), but for arbitrary circuit samplers. This definition clearly leads to problems if the input-output behavior of the sampled circuits C_0 and C_1 is easily distinguishable. Therefore, [ABG+13] restrict the definition of dIO to classes of circuits, where given (C_0, C_1, z) sampled from D, it is infeasible to find an input x, where C_0 and C_1 differ. Another possibility is to require that for every adversary, there exists a suitable extractor recovering an input x, where the sampled circuits differ [BCP14]. The authors of [BCP14; ABG+13; BP15b] provide several applications of dIO, e. g., functional witness encryption, adaptively secure functional encryption, multiparty non-interactive key-exchange (without setup and only logarithmic-size messages), perfect zero knowledge succinct non-interactive arguments (SNARGs), and many more. However, there is evidence that dIO for all circuits is implausible [GGHW17; BP15b].

2.5.1 Probabilistic Indistinguishability Obfuscation

Indistinguishability obfuscation for probabilistic circuits, or probabilistic indistinguishability obfuscation (pIO), was introduced in [CLTV15]. This notion generalizes the classical notion of IO from [Had00; BGI+12]. Similar to Definition 2.15, pIO is defined relative to a class of circuit samplers $\mathfrak{C}$. A circuit sampler is defined as before, see Section 2.5. Informally, a probabilistic indistinguishability obfuscator allows to obfuscate probabilistic circuits and is secure relative to a circuit sampler D if obfuscations $\mathsf{piO}(C_0)$ and $\mathsf{piO}(C_1)$ of circuits $(C_0, C_1) \leftarrow D$ are indistinguishable. Let $\mathcal{C} = (\mathcal{C}_\lambda)_{\lambda \in \mathbb{N}}$ be an ensemble

of sets $\mathcal{C}_\lambda$ of probabilistic circuits. The set $\mathcal{C}_\lambda$ contains circuits of size λ with input length $n(\lambda)$ expecting (at most) $m(\lambda)$ random bits.

DEFINITION 2.16 (Probabilistic indistinguishability obfuscation for a class of samplers $\mathfrak{C}$, [CLTV15]). A *probabilistic indistinguishability obfuscator (pIO)* for a class of samplers $\mathfrak{C}$ over the probabilistic circuit family $\mathcal{C} = (\mathcal{C}_\lambda)_{\lambda\in\mathbb{N}}$ is a uniform PPT algorithm piO, such that the following properties hold.

CORRECTNESS. For every PPT adversary $\mathcal{A}$, every $C \in \mathcal{C}_\lambda$, the advantage of $\mathcal{A}$, given the description of the circuit C, to distinguish oracle access to the original randomized circuit C from oracle access to $\mathsf{piO}(C)$ without querying the same input more than once, is negligible.

SECURITY WITH RESPECT TO $\mathfrak{C}$. For all circuit samplers $D \in \mathfrak{C}$, for all PPT adversaries $\mathcal{A}$, the advantage

$$\begin{aligned}\mathrm{Adv}^{\text{pio-ind}}_{\mathsf{piO},D,\mathcal{A}}(\lambda) \coloneqq \\ \Big|\Pr\left[(C_0,C_1,z)\leftarrow D_\lambda : \mathcal{A}(1^\lambda, C_0, C_1, z, \mathsf{piO}(1^\lambda, C_0)) = 1\right] \\ -\Pr\left[(C_0,C_1,z)\leftarrow D_\lambda : \mathcal{A}(1^\lambda, C_0, C_1, z, \mathsf{piO}(1^\lambda, C_1)) = 1\right]\Big|\end{aligned}$$

is negligible.

For the purpose of a versatile notion of probabilistic indistinguishability obfuscation, we require security to hold for sampler classes which are broader than the restrictive class of functionally equivalent samplers from Definition 2.14. In fact, the power of pIO lies in the generality of feasible circuit sampler classes.

The broadest class of circuit samplers we consider is the class of dynamic-input indistinguishable circuit samplers, introduced in [CLTV15]. Informally, a circuit sampler D is a dynamic-input indistinguishable circuit sampler if D produces circuit pairs (C_0, C_1) such that even for adversarially chosen inputs x, the distributions induced by $C_0(x)$ and $C_1(x)$ are computationally indistinguishable.

This class of circuit samplers is the broadest conceivable class which still results in a feasible notion of pIO. This follows from the observation that if the output distributions $C_0(x)$ and $C_1(x)$ are efficiently distinguishable for some input x, then so are their obfuscations.

DEFINITION 2.17 (Dynamic-input indistinguishable samplers, [CLTV15]). The class $\mathfrak{C}^{\text{dyn-ind}}$ of *dynamic-input indistinguishable samplers* for a circuit family $\mathcal{C}$ contains all circuit samplers $D = \{D_\lambda\}_{\lambda\in\mathbb{N}}$ for $\mathcal{C}$ such that for every (non-uniform) PPT adversary $\mathcal{A}$, the advantage

$$\mathrm{Adv}^{\text{dyn-ind}}_{D,\mathcal{A}}(\lambda) \coloneqq \left|\Pr\left[\mathrm{Exp}^{\text{dyn-ind}}_{D,\mathcal{A}}(\lambda) = 1\right] - \frac{1}{2}\right|$$

is negligible, where the game $\mathrm{Exp}^{\text{dyn-ind}}_{D,\mathcal{A}}(\lambda)$ is defined in Figure 2.2a.

As observed in [CLTV15], restricting pIO for dynamic-input indistinguishable samplers to the circuit class of deterministic circuits results in differing-inputs obfuscation. More precisely, for every $D \in \mathfrak{C}$, given $(C_0, C_1, z) \leftarrow D$, it must be computationally hard to find an input x such that $C_0(x) \neq C_1(x)$. Hence, pIO for dynamic-input indistinguishable samplers inherits the implausibility result from [GGHW17].

$\mathrm{Exp}^{\text{dyn-ind}}_{D,\mathcal{A}}(\lambda)$

$(C_0, C_1, z) \leftarrow D_\lambda$

$(x, st) \leftarrow \mathcal{A}(C_0, C_1, z)$

$y \leftarrow C_b(x)$ for $b \leftarrow \{0,1\}$

$b' \leftarrow \mathcal{A}(st, C_0, C_1, z, x, y)$

return $b = b'$

(a) Game $\mathrm{Exp}^{\text{dyn-ind}}_{D,\mathcal{A}}(\lambda)$ for the indistinguishability property of dynamic-input samplers.

$\mathrm{Exp}^{\text{sel-ind}}_{D,\mathcal{A}}(\lambda)$

$(x, st) \leftarrow \mathcal{A}(1^\lambda)$

$(C_0, C_1, z) \leftarrow D(1^\lambda)$

$b \leftarrow \{0,1\}$

$y \leftarrow C_b(x)$

$b' \leftarrow \mathcal{A}(st, (C_0, C_1, z), y)$

return $b = b'$

(b) Game $\mathrm{Exp}^{\text{sel-ind}}_{D,\mathcal{A}}(\lambda)$ for the X-indistinguishability property of static-input samplers.

Figure 2.2: Definitions of the security games for dynamic-input indistinguishable samplers (2.2a) and (static-input) X-ind samplers (2.2b).

STATIC-INPUT PIO. The above implausibility suggests the study of a weaker notion. Canetti et al. [CLTV15] introduce a corresponding "static-inputs" notion, see Figure 2.2b for the corresponding game. At first, this definition renders the resulting class of circuit samplers even broader (resulting in an impossible notion). However, Canetti et al. restrict this class of circuit samplers accounting for the number of inputs, where the sampled circuits actually behave differently as follows. Informally, let X be (a lower bound of) the number of inputs, where circuits sampled from D behave differently. Then, a circuit sampler D is in the class of X-ind samplers if for every adversary, the distinguishing advantage in the game defined in Figure 2.2b is upper bounded by $\mathit{negl}(\lambda) \cdot X^{-1}(\lambda)$. Due to [CLTV15], pIO for this class of samplers is achievable from IO and one-way functions.

DEFINITION 2.18 (X-ind sampler, [CLTV15]). Let $X(\lambda)$ be a function upper bounded by 2^λ. The class $\mathcal{C}^{\text{X-ind}}$ of *X-ind samplers* for a circuit family $\mathcal{C}$ contains all circuit samplers $D = (D_\lambda)_{\lambda\in\mathbb{N}}$ for $\mathcal{C}$ such that for all $\lambda \in \mathbb{N}$, there exists a set $\mathcal{X}_\lambda \subseteq \{0,1\}^*$ with $|\mathcal{X}| \leqslant X(\lambda)$, such that

X-DIFFERING INPUTS. With overwhelming probability over the choice of $(C_0, C_1, z) \leftarrow D_\lambda$, for every $x \notin \mathcal{X}_\lambda$, for all $r \in \{0,1\}^{m(\lambda)}$, we have that $C_0(x; r) = C_1(x; r)$.

X-INDISTINGUISHABILITY. For all (non-uniform) adversaries $\mathcal{A}$, the advantage

$$X(\lambda) \cdot \left(\Pr\left[\mathrm{Exp}^{\text{sel-ind}}_{D,\mathcal{A}}(\lambda) = 1\right] - \frac{1}{2} \right)$$

is negligible, where $\mathrm{Exp}^{\text{sel-ind}}_{D,\mathcal{A}}(\lambda)$ is defined in Figure 2.2b.

On a high level, the construction of pIO for the sampler class $\mathcal{C}^{\text{X-ind}}$ from [CLTV15] works as follows. Given a probabilistic circuit C with size at most λ, $\mathsf{piO}(C)$ samples a PRF key K and obfuscates the deterministic circuit $\overline{C}$ which on input of x evaluates the deterministic circuit $C(x; F(K, x))$.

This construction respects the support of the original randomized circuit, i. e., the output of the obfuscated circuit is always in the support of the original circuit (on the same inputs). More formally, for all circuits $C \in \mathcal{C}_\lambda$,

all inputs $x \in \{0,1\}^*$ (of matching length), all $\Lambda \in \mathit{supp}(\mathsf{piO}(C))$, we have $\Lambda(x) \in \mathit{supp}(C(x))$.

NOTIONS OF CORRECTNESS. Defining correctness for probabilistic obfuscation is slightly more tricky than for deterministic obfuscation. This is because an obfuscated program is defined to be *deterministic* whereas the original program can use randomness. In [CLTV15], correctness is defined by requiring that evaluations of the obfuscated program must be indistinguishable from evaluations of the original randomized program, where the randomized program must not be called twice on the same input (to exclude trivial attacks). This definition of correctness is inspired by the security definition a pseudorandom function, see Definition 2.10, which is used by [CLTV15] to derive the randomness for the obfuscated circuit.

Another notion of correctness is to require that the obfuscated program *respects the support* of the original randomized program. This notion is useful because, in many cases, it allows to avoid a reduction to the correctness property above which can simplify proofs.

Dodis et al. [DHRW16] introduce another notion of correctness which guarantees that a one-time evaluation of the obfuscated program is distributed *identically* to the distribution induced by the original randomized circuit (on the same input), where the distribution is also over the randomness of the obfuscator. Clearly, this notion of correctness implies that the obfuscated circuit respects the support of the original program.

In Part I, we elaborate on these notions in more detail and, in Part II, we introduce a novel notion of *statistical* correctness which proves to be quite useful.

2.6 ASSUMING ADVERSARIAL KNOWLEDGE

To make assumptions on adversarial knowledge is a common paradigm used in cryptography. Such assumptions state that if an adversary is able to derive some output, then he *must know* how he computed that output. This kind of "adversarial knowledge" is modeled by the ability to algorithmically extract certain information from the adversary's code (and random tape).

With the knowledge-of-exponent assumption (KEA), Damgård [Dam92] introduced the first assumption about adversarial knowledge. The KEA is an assumption on cryptographic groups and informally states that given two random group generators g and h, then the only way to produce (A, B) such that $\mathit{dlog}_A(B) = \mathit{dlog}_g(h)$ is to compute (g^c, h^c) for some $c \in \mathbb{Z}_p$. That is, every adversary producing such (A, B) must know c such that $(A, B) = (g^c, h^c)$.

DEFINITION 2.19 (Knowledge-of-exponent assumption, [Dam92]). Let $\mathbb{G}$ be a cyclic group with group parameter generation algorithm GGen producing triplets of the form $pp_{\mathbb{G}} := (\mathbb{G}, p, g)$. The knowledge-of-exponent assumption

(KEA) holds relative to GGen if for every PPT adversary $\mathcal{A}$, there exists a PPT extractor $\mathcal{E}$ such that

$$\Pr\left[\begin{array}{rcl} pp_G & \leftarrow & \mathsf{GGen}(1^\lambda) \\ x & \leftarrow & \mathbb{Z}_p \\ (A,B) & \leftarrow & \mathcal{A}(pp_G, g^x; r) \\ c & \leftarrow & \mathcal{E}(pp_G, g^x, r) \end{array} : A^x = B \wedge g^c \neq A\right]$$

is a negligible function in λ.

Note that KEA is only nontrivial if computing discrete logarithms is hard relative to GGen.

This paradigm of knowledge-type assumptions was later generalized as extractable functions by Canetti and Dakdouk [CD08; CD09]. These are (one-way) function families such that every adversary who (given a function description f) outputs an image $y \in \mathit{image}(f)$ must know a pre-image of y under f.

DEFINITION 2.20 (Extractable one-way functions without auxiliary information, [BCPR16]). An *extractable one-way function (EOWF) family ensemble without auxiliary information* is a function family ensemble $\mathcal{F} = (\mathsf{Gen}, \mathsf{Eval})$ (cf. Definition 2.8) if the following two properties are satisfied.

ONE-WAYNESS. For every PPT adversary $\mathcal{A}$,

$$\Pr\left[\mathrm{Exp}^{\mathsf{ow}}_{\mathcal{F},\mathcal{A}}(\lambda) = 1\right]$$

is negligible, where $\mathrm{Exp}^{\mathsf{ow}}_{\mathcal{F},\mathcal{A}}(\lambda)$ is defined in Figure 2.3a.

EXTRACTABILITY. For every PPT adversary $\mathcal{X}$ (using a random tape of length $p(\lambda)$), there exists a PPT algorithm $\mathcal{E}_\mathcal{X}$ such that

$$\Pr\left[\mathrm{Exp}^{\mathsf{ext}}_{\mathcal{F},\mathcal{X},\mathcal{E}_\mathcal{X}}(\lambda) = 1\right]$$

is overwhelming, where $\mathrm{Exp}^{\mathsf{ext}}_{\mathcal{F},\mathcal{X},\mathcal{E}_\mathcal{X}}(\lambda)$ is defined in Figure 2.3b.

$\mathrm{Exp}^{\mathsf{ow}}_{\mathcal{F},\mathcal{A}}(\lambda)$

$f \leftarrow \mathsf{Gen}(1^\lambda)$
$x \leftarrow \{0,1\}^{n(\lambda)}$
$y \coloneqq f(x)$
$x' \leftarrow \mathcal{A}(f, y)$
return $f(x') = y$

(a) The one-way game.

$\mathrm{Exp}^{\mathsf{ext}}_{\mathcal{F},\mathcal{X},\mathcal{E}_\mathcal{X}}(\lambda)$

$f \leftarrow \mathsf{Gen}(1^\lambda)$
$r_\mathcal{X} \leftarrow \{0,1\}^{p(\lambda)}$
$y \leftarrow \mathcal{X}(f; r_\mathcal{X})$
$x \leftarrow \mathcal{E}_\mathcal{X}(f, r_\mathcal{X})$
return $\big(f(x) = y\big) \vee \big(\forall x' \colon f(x') \neq y\big)$

(b) The extraction game.

Figure 2.3: One-way and extraction games for EOWFs.

DEFINITION 2.21 (Extractable one-way function family ensembles with common auxiliary information, [BCPR16]). An *extractable one-way function (EOWF) family ensemble with common auxiliary information* is a function family ensemble $\mathcal{F} = (\mathsf{Gen}, \mathsf{Eval})$ (cf. Definition 2.8) if the following properties are satisfied.

ONE-WAYNESS. As in Definition 2.20.

ONE-WAYNESS (STRONGER). For every PPT adversary $\mathcal{A}$, for every polynomial b and for every $z \in \{0,1\}^{b(\lambda)}$,

$$\Pr\left[\mathrm{Exp}^{\text{ow-aux}}_{\mathcal{F},\mathcal{A},z}(\lambda) = 1\right]$$

is negligible, where $\mathrm{Exp}^{\text{ow-aux}}_{\mathcal{F},\mathcal{A},z}(\lambda)$ is defined in Figure 2.4a.

EXTRACTABILITY. For every PPT adversary $\mathcal{X}$, there exists a PPT algorithm $\mathcal{E}_{\mathcal{X}}$ such that for every polynomial b and every $z \in \{0,1\}^{b(\lambda)}$,

$$\Pr\left[\mathrm{Exp}^{\text{ext-aux}}_{\mathcal{F},\mathcal{X},\mathcal{E}_{\mathcal{X}},z}(\lambda) = 1\right]$$

is overwhelming, where $\mathrm{Exp}^{\text{ext-aux}}_{\mathcal{F},\mathcal{X},\mathcal{E}_{\mathcal{X}},z}(\lambda)$ is defined in Figure 2.4b.

$\mathrm{Exp}^{\text{ow-aux}}_{\mathcal{F},\mathcal{A},z}(\lambda)$

$f \leftarrow \mathsf{Gen}(1^\lambda)$
$x \leftarrow \{0,1\}^{n(\lambda)}$
$y \coloneqq f(x)$
$x' \leftarrow \mathcal{A}(f, y, z)$
return $f(x') = y$

(a) The one-way game.

$\mathrm{Exp}^{\text{ext-aux}}_{\mathcal{F},\mathcal{X},\mathcal{E}_{\mathcal{X}},z}(\lambda)$

$f \leftarrow \mathsf{Gen}(1^\lambda)$
$r_{\mathcal{X}} \leftarrow \{0,1\}^{p(\lambda)}$
$y \leftarrow \mathcal{X}(f, z; r_{\mathcal{X}})$
$x \leftarrow \mathcal{E}_{\mathcal{X}}(f, z, r_{\mathcal{X}})$
return $\big(f(x) = y\big) \vee \big(\forall x' \colon f(x') \neq y\big)$

(b) The extraction game.

Figure 2.4: One-way and extraction game for EOWFs with common auxiliary input.

We note that the weaker notion of one-wayness without auxiliary input is sufficient for us. Assuming non-uniform adversaries, one-wayness without auxiliary input and one-wayness with auxiliary input are equivalent. We emphasize that in contrast to one-wayness, extractability with common auxiliary input is strictly stronger than extractability without common auxiliary input (even when considering non-uniform adversaries).

DEFINITION 2.22 (Extractable one-way function family ensembles with b-bounded common auxiliary information, [BCPR16]). Like Definition 2.21 but with a fixed polynomial b determining the length of the common auxiliary information.

In Section 17.2.1 we define further variants of EOWFs.

ON THE NEED FOR NON-BLACK-BOX TECHNIQUES. The knowledge-of-exponent assumption [HT98; BP04] as well as extractable one-way functions [CD09; BCC+17] imply 3-message zero-knowledge protocols. However, this notion of zero-knowledge protocols is known to be impossible with respect to black-box simulation [GK96; Kat08; FGJ18]. Hence, constructing EOWFs (or groups where KEA holds) requires non-black-box techniques. So far, no such constructions are known.[22]

[22] *Note that Barak [Bar01] introduces a non-black-box technique that does not rely on knowledge-type assumptions. However, it is not known whether this technique extends to settings other than zero-knowledge proofs.*

FALSIFIABILITY AND PLAUSIBILITY. Furthermore, knowledge-type assumptions such as the knowledge-of-exponent assumption or the existence of extractable one-way functions are not falsifiable in the sense of Naor [Nao03],

see Section 2.2.1. Hence, it is difficult to reason about their plausibility. While assuming KEA or the existence of EOWFs might be better than simply assuming "the scheme is secure", a proof based on such assumptions is not satisfying.

In Part III of this thesis, we use knowledge-type assumptions "negatively" in the sense that we prove that the existence of certain knowledge-type assumptions conflicts with other notions. Furthermore, in Part II of this thesis, we demonstrate how certain knowledge-type assumptions can be avoided.

We note that the notion of extractable one-way functions with unbounded common auxiliary information as per Definition 2.21 conflicts with indistinguishability obfuscation [BCPR16] since the common auxiliary information can contain a circuit which, given a function description f, outputs $f(\mathsf{F}(\mathsf{K}, f))$ thus forcing any extractor to break the pseudorandom function F. More precisely, consider the universal adversary which simply executes all commands in its auxiliary input. Then, there needs to be one "universal" extractor which works for *all* adversarial strategies. Suppose the auxiliary input contains an *obfuscated* program which computes $f(\mathsf{F}(\mathsf{K}, f))$ as above, then the universal extractor needs to either break the pseudorandom function F or the security of the used obfuscator. Hence, it is unlikely that such EOWFs exist (even heuristically).

2.7 PUBLIC-KEY ENCRYPTION AND ITS VARIANTS

Encryption enables users to confidentially communicate messages over insecure channels. Secret-key encryption (SKE) assumes both users know a common secret key which is used for both encryption and decryption. SKE is known to exist from OWFs [GM84]. In contrast to classical secret-key encryption, public-key encryption (PKE) schemes use two keys; a public key for encryption and a secret key for decryption. Currently, it is not known whether PKE schemes can be constructed from OWFs but there is evidence, that this might not be possible [IR89; Dac16]. However, if IO exists, then PKE schemes can be constructed from OWFs.

DEFINITION 2.23 (Public-key encryption, [ElG85; Yao82]). A *public-key encryption (PKE) scheme* for message space $\mathcal{M} = \{\mathcal{M}_\lambda\}_{\lambda\in\mathbb{N}}$ is a tuple of algorithms $\mathsf{E} = (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ such that the following properties are satisfied.

PERFECT CORRECTNESS. For all $\lambda \in \mathbb{N}$, all $(pk, sk) \in \mathit{supp}(\mathsf{KGen}(1^\lambda))$, all $m \in \mathcal{M}_\lambda$,

$$\Pr\left[\mathsf{Dec}(sk, \mathsf{Enc}(pk, m)) = m\right] = 1.$$

IND-CPA SECURITY. For all legitimate PPT adversaries $\mathcal{A}$,

$$\mathrm{Adv}^{\mathsf{ind\text{-}cpa}}_{\mathsf{E},\mathcal{A}}(\lambda) \coloneqq \left|\Pr\left[\begin{array}{rcll} (pk, sk) & \leftarrow & \mathsf{KGen}(1^\lambda) & \\ (m_0, m_1, st) & \leftarrow & \mathcal{A}(pk) & : \mathcal{A}(pk, c^*, st) = 1 \\ c^* & \leftarrow & \mathsf{Enc}(pk, m_0) & \end{array}\right]\right.$$

$$-\Pr\left[\begin{array}{rcll}(pk,sk) & \leftarrow & \mathsf{KGen}(1^\lambda) & \\ (m_0,m_1,st) & \leftarrow & \mathcal{A}(pk) & : \mathcal{A}(pk,c^*,st)=1 \\ c^* & \leftarrow & \mathsf{Enc}(pk,m_1) & \end{array}\right]\Bigg|$$

is negligible, where legitimate means that $\mathcal{A}$ always outputs two messages $m_0, m_1 \in \mathcal{M}$ of identical length.

Without loss of generality, we assume that sk is the random tape used for key generation. Therefore, making the random tape of KGen explicit, we write $(pk,sk) = \mathsf{KGen}(1^\lambda; sk)$.

Note that IND-CPA security as defined in Definition 2.23 is not the de facto security for use in practice. However, it suffices as building block for our applications.

2.7.1 Fully Homomorphic Encryption

Equipping public-key encryption schemes with the additional property that one can publicly evaluate programs on the contents of ciphertexts was a long standing open problem since [RAD78] and was resolved by Gentry [Gen09]. This variant of public-key encryption has many applications in theory and practice. For instance, fully homomorphic encryption allows to outsource computations on private data without compromising its confidentiality.

Let $\mathcal{P} = (\mathcal{P}_\lambda)_{\lambda\in\mathbb{N}}$ be a family of sets of polynomial sized circuits of arity $a(\lambda)$, i. e., the set $\mathcal{P}_\lambda$ contains circuits of polynomial size in λ. We assume that for any $\lambda \in \mathbb{N}$ the circuits in $\mathcal{P}_\lambda$ share the common input domain $(\{0,1\}^{poly(\lambda)})^{a(\lambda)}$ for a fixed polynomial $poly(\lambda)$. A homomorphic encryption scheme enables evaluation of circuits on encrypted data.

DEFINITION 2.24 (Fully homomorphic encryption, [Gen09]). A *fully homomorphic (public-key) encryption (FHE) scheme* with message space $\mathcal{M} = \{\mathcal{M}_\lambda\}_{\lambda\in\mathbb{N}}$ for a deterministic circuit family $\mathcal{P} = (\mathcal{P}_\lambda)_{\lambda\in\mathbb{N}}$ of arity $a(\lambda)$ and input domain $(\{0,1\}^{poly(\lambda)})^{a(\lambda)}$ is a tuple of PPT algorithms $\mathsf{FHE} = (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ such that $(\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ is a perfectly correct IND-CPA secure public-key encryption scheme and the following properties are met.

PERFECT CORRECTNESS. For all $\lambda \in \mathbb{N}$, all $m_1, \ldots, m_{a(\lambda)} \in \mathcal{M}_\lambda$, $P \in \mathcal{P}_\lambda$,

$$\Pr\left[\begin{array}{rcll}(pk,sk) & \leftarrow & \mathsf{KGen}(1^\lambda) & \\ c_i & \leftarrow & \mathsf{Enc}(pk,m_i) & : \\ c & \leftarrow & \mathsf{Eval}\big(pk,P,c_1,\ldots,c_{a(\lambda)}\big) & \end{array}\begin{array}{c}\mathsf{Dec}(sk,c) \\ = \\ P\big(m_1,\ldots,m_{a(\lambda)}\big)\end{array}\right] = 1.$$

COMPACTNESS. The size of the output of Eval is polynomial in λ and independent of the size of the circuit P.

The first FHE scheme based on standard lattice assumptions (additionally assuming circular security) is due to [BV14a; BV11; BV14b].[23]

Further, due to [CLTV15], probabilistic indistinguishability obfuscation in conjunction with (slightly super-polynomially secure) perfectly correct and perfectly re-randomizable public-key encryption yields a perfectly correct and perfectly re-randomizable fully homomorphic encryption scheme (without using circular security assumptions).

[23] *Circular security means that it is assumed that the scheme remains secure even if an encryption of the secret key is known.*

2.8 NON-INTERACTIVE ZERO-KNOWLEDGE PROOF SYSTEMS

Zero-knowledge protocols [GMR85] are a fundamental notion in cryptography. They allow to prove the validity of statements without revealing any information beyond that fact. One-way functions are both sufficient [GMR85] and necessary [OW93] for the existence of interactive zero-knowledge protocols for languages in $\mathcal{NP}$.

Non-interactive zero-knowledge (NIZK) proof systems [BFM88] are the non-interactive analogue of zero-knowledge protocols, where only one message between the prover and the verifier is exchanged. In contrast to interactive zero-knowledge protocols, NIZK proof systems in the plain model are only possible for languages in $\mathcal{BPP}$ [GO94]. However, the notion of zero-knowledge is not meaningful if restricted to $\mathcal{BPP}$ since a verifier does not need the prover to verify the validity of a statement. Hence, NIZK proof systems for $\mathcal{NP}$ only exist in stronger models such as the common reference string model, where a reference string is generated by a trusted party and made available to all parties.

NIZK proof systems find use in various areas of cryptography. A NIZK proof system for a language L with witness relation $\mathcal{R}_L$ enables to prove in a non-interactive manner that some statements are in L without leaking information about corresponding witnesses.

DEFINITION 2.25 (Non-interactive zero-knowledge proof system [GOS06]). A *non-interactive zero-knowledge (NIZK) proof system* for a language $L \in \mathcal{NP}$ (with witness relation $\mathcal{R}_L$) is a tuple of PPT algorithms $\mathsf{NIZK} = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ such that Setup is a common reference string generation algorithm, Prove is a proving algorithm and Verify is a (deterministic) verification algorithm.

- $\mathsf{Setup}(1^\lambda)$ outputs a common reference string σ.
- $\mathsf{Prove}(\sigma, x, w)$, on input of σ, a statement x and a witness w, outputs a proof π.
- $\mathsf{Verify}(\sigma, x, \pi)$, on input of σ, a statement x and a proof π, deterministically outputs either 1 or 0.

A NIZK proof system is required to meet the following properties.

PERFECT COMPLETENESS. For every $\lambda \in \mathbb{N}$, every $(x, w) \in \mathcal{R}_L$ with $|x| = \lambda$, we have that

$$\Pr\left[\begin{array}{l}\sigma \leftarrow \mathsf{Setup}(1^\lambda) \\ \pi \leftarrow \mathsf{Prove}(\sigma, x, w)\end{array} : \mathsf{Verify}(\sigma, x, \pi) = 1\right] = 1,$$

where the probability is over the random coins of Setup and Prove.

STATISTICAL SOUNDNESS. For every $\lambda \in \mathbb{N}$, every $x \notin L$ (of polynomial length) and every (possibly unbounded) adversary $\mathcal{A}$,

$$\begin{aligned}\mathrm{Adv}^{\mathsf{sound}}_{\mathsf{NIZK},\mathcal{A}}(\lambda) &:= \Pr\left[\begin{array}{l}\sigma \leftarrow \mathsf{Setup}(1^\lambda) \\ \pi \leftarrow \mathcal{A}(\sigma, x)\end{array} : \mathsf{Verify}(\sigma, x, \pi) = 1\right] \\ &< 2^{-\lambda},\end{aligned}$$

where the probability is over the random coins of Setup and $\mathcal{A}$.

COMPUTATIONAL ZERO-KNOWLEDGE. There exists a PPT algorithm $\mathsf{Sim} = (\mathsf{Sim}_0, \mathsf{Sim}_1)$ such that for every PPT adversary $\mathcal{A}$,

$$\begin{aligned}&\mathrm{Adv}^{\mathsf{zk}}_{\mathsf{NIZK},\mathsf{Sim},\mathcal{A}}(\lambda) \coloneqq \\ &\left| \Pr\left[\sigma \leftarrow \mathsf{Setup}(1^\lambda) \colon \mathcal{A}^{\mathsf{Prove}(\sigma,\cdot,\cdot)}(\sigma) = 1\right]\right. \\ &\left. - \Pr\left[(\sigma, \tau_\Pi) \leftarrow \mathsf{Sim}_0(1^\lambda) \colon \mathcal{A}^{\mathsf{Sim}'_1(\sigma,\tau_\Pi,\cdot,\cdot)}(\sigma) = 1\right] \right|\end{aligned}$$

is negligible in λ, where $\mathsf{Sim}'_1(\sigma, \tau_\Pi, x, w)$ operates like $\mathsf{Sim}_1(\sigma, \tau_\Pi, x)$ only if $(x, w) \in \mathcal{R}_L$ and returns $\perp$ otherwise.

REMARK 2.1. The above definition of NIZK proof systems can be strengthened such that NIZK.Setup is required to produce uniformly random common reference strings from $\{0,1\}^{n_\Pi(\lambda)}$, so-called common *random* strings.

For interactive zero-knowledge proof systems, one-way functions are sufficient [GMW86] (even in the plain model, i. e., without a common reference string). This is not the case for non-interactive zero-knowledge proof systems. To date, NIZK proof systems can be built from a variety of assumptions including assumptions on pairing-friendly groups [GOS06; CH19], certain trapdoor permutations (and, hence, RSA) [FLS90], the learning-with-errors (LWE) assumption [PS19] and indistinguishability obfuscation (in conjunction with one-way functions) [BP15a].

Part I

DOUBLY-PROBABILISTIC INDISTINGUISHABILITY OBFUSCATION

3 INTRODUCTION

In this part, we present the results of [ACH20]. This part is taken verbatim from [ACH20] with minor changes.

OBFUSCATION. Code obfuscation has been formalized already in the early 2000s as a cryptographic building block, by Hada [Had00] and Barak et al. [BGI+01], along with a number of early positive [Can97; LPS04; Wee05; HRsV07; HMS07] and negative [BGI+01; GK05; Wee05] results. However, prior to the candidate obfuscation scheme of Garg et al. [GGH+13], only relatively few positive results on obfuscation were known.

The first candidate obfuscator from [GGH+13] changed things. Their work identified indistinguishability obfuscation (IO, cf. [BGI+01; GR07]) as an achievable *and* useful general notion of obfuscation: it presented a candidate indistinguishability obfuscator, along with a first highly nontrivial application (functional encryption). Since then, a vast number of applications have been proposed, ranging from functional [GGH+13], deniable [SW14], and fully homomorphic [CLTV15] encryption, over multi-party computation (e. g., [GGHR14]), to separation results (e. g., [HRW16]). In the process, powerful techniques like "puncturing" [SW14] have been discovered, which have found applications even beyond obfuscation (e. g., in multi-party computation [BL18a; GS18], instantiating the Fiat-Shamir paradigm [CCRR18], and verifiable random functions [Bit17; GHKW17]). Besides, the notion of IO itself has been refined, and related to other notions of obfuscation [ABG+13; BP13; BCP14; BCKP14; CLTV15; IPS15], and various different constructions of obfuscators have been presented [PST14; Zim15; AJ15; BV15; AS17; Lin17; LT17; CVW18; GJK18; CHVW19; AJL+19; Agr19; JLMS19; BDGM20a; GJLS20; JLS20; GP20; AP20; BIJ+20; BDGM20b; WW20].

SUBEXPONENTIAL ASSUMPTIONS. It is currently hard to find a cryptographic primitive that can *not* be constructed from IO (in combination with another mild assumption such as the existence of one-way functions). However, some of the known IO-based constructions come only with *subexponential* reductions to IO. For instance, the only known IO-based constructions of fully homomorphic encryption [CLTV15], spooky encryption [DHRW16], and graded encoding schemes [FHHL18] suffer from reductions with a subexponential loss.

Hence, while IO has generally been recognized as an extremely powerful primitive (even to the extent being called a "central hub" for cryptography [SW14]), it is not at all clear if this also holds for *polynomially* secure IO. Indeed, it is conceivable that only polynomially secure IO exists, in which case much of IO's power stands in question.

More generally, subexponential reductions (in particular to IO) are undesirable. Namely, the security of existing IO constructions is still not well-understood, and in particular current state-of-the-art constructions of IO schemes (such as [AS17; Lin17; LT17]) already require subexponential com-

putational assumptions themselves. Hence, assuming subexponential IO is a particularly risky bet. This suspicion is confirmed in part by Pass and shelat [Ps16] who separate polynomial and subexponential security for virtual black-box obfuscation.

Removing subexponential assumptions in general and from IO-based constructions in particular has already explicitly been considered in [LM16; GS16] and [GPS16; GPSZ17; LZ17] respectively. These works offer general techniques and ideas to turn subexponential reductions into polynomial ones. For instance, [GPSZ17; LZ17] offer ways to replace (subexponential) IO-based constructions with (polynomial) constructions based on functional encryption. Of course, this requires a special structure of the primitive to be implemented, and is demonstrated for several primitives, including non-interactive key exchange and short signature schemes.

OUR CONTRIBUTION. In this part, we are also concerned with substituting subexponential with polynomial reductions in IO-based constructions. Unlike [GPSZ17; LZ17], however, we do not follow the approach of using functional encryption directly in place of IO, but instead will employ extremely lossy functions (ELFs) [Zha16] to "absorb" subexponential complexity.[24]

We will implement a variant of probabilistic indistinguishability obfuscation (pIO, introduced in [CLTV15]) using polynomially secure IO (and ELFs). A pIO scheme can be used to obfuscate *probabilistic* (i. e., randomized) programs, and are currently the only way to obtain, e. g., fully homomorphic encryption (FHE) schemes without circular security assumptions [CLTV15]. However, the only previous construction of pIO schemes required subexponentially secure IO [CLTV15]. Hence, our construction yields the first FHE scheme from polynomially secure IO (and ELFs). Similarly, we can turn the assumption of subexponentially secure IO into polynomially secure IO (plus ELFs) in the construction of spooky encryption from [DHRW16].

Both FHE and spooky encryption are quite powerful primitives, and we obtain several "spin-off results" by revisiting their implications. For instance, when instantiating the pIO-based FHE construction of [CLTV15] with our pIO scheme and a suitable public-key encryption scheme, we obtain a fully key-dependent message (KDM) secure public-key encryption scheme from (polynomially secure) IO and the exponentially secure DDH assumption (and no further assumptions). Under the same assumptions, we obtain multi-key FHE with threshold decryption and function secret sharing schemes from the spooky encryption construction from [DHRW16].

[24] *That means that our final schemes depend on ELFs, which are currently only known to be instantiable from exponential assumptions. However, we stress that ELFs can be built from exponential variants of very standard assumptions, such as the decisional Diffie-Hellman (DDH) assumption.*

ON THE PLAUSIBILITY OF ELFS. One could argue that we trade one exponential assumption for another, and it is not clear that assuming polynomial IO and exponential DDH is any better than assuming only subexponential IO in the first place. Seconding Zhandry [Zha16] here, we think that exponential DDH is a realistic assumption that is far more popular, better-investigated, and arguably more plausible than subexponential IO. Much of the currently deployed cryptography relies on (in fact a strong variant of) exponential DDH, because parameters are almost always chosen according to the best known attacks.

ON THE NUMBER OF ASSUMPTIONS. Another natural observation is that IO for general circuits is already an exponential family of assumptions in

itself (one for each obfuscated circuit) [PST14]. It might seem that this lets the challenge of relying on polynomially secure IO instead of subexponentially secure IO appear less appealing. We make two comments on that.

- First, being an exponential family of assumptions and assuming resistance against subexponential adversaries are orthogonal issues. Many cryptographic assumptions have several dimensions of strengths, and relaxing the assumption in any of these dimensions is desirable.[25] In this part, we make progress in one important dimension. By replacing subexponential IO by polynomial IO plus exponential DDH, we effectively trade an *exponential* number of subexponential hardness assumptions in exchange for a *single* (plausible, well-studied) exponential hardness assumption (plus an exponential family of polynomial hardness assumptions).
- Second, IO being an exponential family of assumptions can be considered an artificial consequence of working on the general notion of IO for *arbitrary circuits*. When using IO in concrete constructions (e. g., in all the constructions described in this part), one almost never needs to assume IO for all circuits. It usually suffices to assume IO for a constant number of specific circuits (namely those being obfuscated in the construction and the analysis). Hence, IO is a small number of assumptions when used for building a cryptographic primitive.

[25] *For example, if a protocol relies on the subexponential hardness of LWE with exponential modulus-to-noise ratio, it would be desirable to achieve the same while relying only on polynomially secure LWE, even if the modulus-to-noise ratio remains exponential.*

3.1 TECHNICAL OVERVIEW

THE PIO CONSTRUCTION OF CANETTI ET AL. To describe our ideas, it will be helpful to briefly review the work of Canetti et al. [CLTV15]. In a nutshell, they define the notion of pIO as a way to obfuscate probabilistic programs, and show how to use pIO to implement the first FHE scheme without any circular security assumption. Intuitively, where the notion of IO captures that the obfuscation $\mathsf{iO}(P)$ of a *deterministic* program P does not leak anything beyond the functionality of P, pIO captures the same for *probabilistic* programs P.[26]

They also show how to implement pIO with an indistinguishability obfuscator iO and a pseudorandom function (PRF) F. Namely, in order to obfuscate a probabilistic program P, Canetti et al. obfuscate the *deterministic* program P' that, on input x, runs $P(x)$ with random coins $r = F(K, x)$. Here, K is a PRF key hard-coded into P'. The security proof uses "puncturing" techniques [SW14] and a hybrid argument over all possible P-inputs x. More specifically, for each P-input x, separate reductions to the security of iO and F show that the execution of $P'(x)$ is secure.[27]

This proof strategy is very general and does not need to make any specific assumptions about the structure of P. (In fact, this strategy can be viewed as a specific form of "complexity leveraging", technically similar to the conversion of selective security into adaptive security, e. g., [BB04a].) However, the price to pay is a reduction loss which is linear in the size of the input domain (which usually is exponentially large). In particular, even after scaling security parameters suitably, Canetti et al. still require subexponentially secure IO and PRFs.

[26] *This is of course an oversimplification. Also, [CLTV15] define several types of pIO security that provide a tradeoff between security and achievability.*

[27] *Again, we are not very specific about the form of desired or assumed security. However, we believe that for this exposition, these specifics do not matter.*

MORE ON PREVIOUS WORKS TO REMOVE SUBEXPONENTIALITY. There are a number of known ways to deal with subexponential reduction losses due to complexity leveraging (or related techniques). For instance, various semi-generic (pre-IO) techniques seek to achieve adaptive security (for different primitives) by establishing an algebraic or combinatorial structure on the used inputs [BB04b; Wat05; HK08; HW09], and can sometimes be adapted to the IO setting [HSW14]. But like the already-mentioned, somewhat more general approaches [GPSZ17; LZ17], these works make specific assumptions about the structure of the involved computations.

A somewhat more general approach (that works for more general classes of programs) was outlined by Zhandry [Zha16], who introduces the notion of "extremely lossy functions" (ELFs). Intuitively, an ELF is an injective function G that can be switched into an "extremely lossy mode", in which its range is polynomially small. Such an ELF can sometimes be used to "preprocess" inputs in a cryptographic scheme, with the following benefit: a security reduction can switch the ELF to extremely lossy mode, so that only a polynomial number of (preprocessed) inputs $G(x)$ need to be considered. This simplifies a potential hybrid argument over all (preprocessed) inputs $G(x)$, and can lead to a polynomial (instead of a subexponential) reduction.

However, trying to apply this strategy to the construction and reduction of Canetti et al. (as sketched above) directly fails. Namely, in their application, inputs to the cryptographic scheme will be inputs x to an arbitrary (probabilistic) program P; preprocessing them with an ELF will destroy their structure, and it is not clear how to run P on ELF-preprocessed inputs $G(x)$. Indeed, applying ELFs to realize pIO requires fundamentally different techniques.

MAIN IDEA: PIO WITH SPARSIFIABLE INPUTS. Instead, we will restrict ourselves to programs P that take as input an element x from a small number of (arbitrary but efficiently samplable) distributions. In other words, all possible inputs x need to be in the range of one of a small number of efficient samplers S_i. As an example, for $i \in \{0,1\}$, sampler S_i could sample ciphertexts C that encrypt plaintext i. Moreover, we require that all inputs to a program P to be obfuscated are at some point actually sampled from some S_i according to a certain process.

Obfuscating a given probabilistic program P (that takes as inputs one or more x as above) now consists of two steps:

1. First, we *encode* all inputs x, in the sense that we compile S_i to attach a "certificate" *aux* to x. This certificate *aux* guarantees that x has really been sampled using S_i. Furthermore, the compiled sampler S_i uses preprocessed random coins of the form $G(r)$ (instead of r) for an ELF G. (When G is in injective mode, this does not affect the distribution of sampled x.) The certificate *aux* additionally guarantees this choice of random coins.[28]

2. Second, we produce the actual obfuscation of the probabilistic program P as follows. We use an indistinguishability obfuscator iO to obfuscate the following (deterministic) variant P' of P: on inputs $x_1, \ldots, x_\ell$ with certificates $aux_1, \ldots, aux_\ell$, P' first checks the certificates aux_i and aborts if one of them is invalid. Next, P' runs $P(x_1, \ldots, x_\ell)$, with random

[28] *Looking ahead, this "certificate" will be implemented using a NIZK in our construction.*

coins $F(K, (x_i)_{i=1}^{\ell})$ for a PRF F and a hard-coded PRF key K. Finally, P′ outputs P's output.

Maybe the most important property of this setup is that now the sets of inputs x_i are "sparsifiable" in the following sense. If we set G to extremely lossy mode, then only a polynomial number of different random coins r can occur. Hence, each S_i will output one of only a small number of possible samples (e. g., encryptions C generated with random coins from a small set). In that sense, the set of possible inputs x_i to P has been "sparsified", and a hybrid argument over all possible inputs as in [CLTV15] is possible with polynomial loss.

We stress that our technique of applying ELFs fundamentally differs from [Zha16]. In [Zha16], the constructed primitive itself ensures that G is applied on all inputs. When approaching the challenge of constructing pIO, however, the input to the primitive must externally be sampled using random coins that are preprocessed with G. This process is not under the control of the primitive and therefore requires a mechanism certifying that inputs are generated according to this specific process. We implement this mechanism using the combination of compiling the sampler for the input distribution into a "certifying sampler" (step 1) and restricting correctness of the obfuscated program (step 2).

Surprisingly, our pIO scheme achieves the notion of "dynamic-input pIO" [CLTV15], a very strong variant of pIO security. On a high level, dynamic-input pIO guarantees indistinguishability between obfuscations of probabilistic programs as long as their output distributions on adversarially chosen inputs are indistinguishable. This constitutes a very strong requirement and, in fact, implies differing-inputs obfuscation [BGI+01; ABG+13], a notion for which strong impossibility results exist [GGHW17; BSW16]. However, our obfuscator produces circuits which are only required to work on inputs certifiably generated according to a specific process. Hence, our pIO scheme enjoys a restricted form of correctness. This enables us to circumvent the impossibility results [GGHW17; BSW16].

APPLICATIONS. One obvious question is of course how restrictive our assumption on input domains really is. We show that our assumptions apply to two existing pIO-based constructions, with a number of interesting consequences.

First, we revisit the pIO-based construction of fully homomorphic encryption from [CLTV15]. Here, pIO is used to obfuscate the FHE evaluation algorithm that takes two ciphertexts (say, of two bit plaintexts b_0 and b_1) as input, and outputs a ciphertext of the NAND of the two plaintexts (i. e., $b_0 \overline{\wedge} b_1$). If we set S_b to be a sampler that samples an encryption of b, this setting perfectly fits our scheme. Hence, we obtain first a leveled homomorphic encryption (LHE) scheme, and from this an FHE scheme using the high-level strategy from [CLTV15]. Hence, putting this together with our pIO construction, we obtain an FHE scheme from polynomially secure IO and an ELF (and no further assumptions).

We note that the above FHE scheme is also fully key-dependent message (KDM, see [BRS03]) secure when implemented with a suitable basic public-key encryption scheme (such as the DDH-based scheme of [BHHO08]). In that case, the FHE is secure even when an encryption of its own secret key $C_{sk} = \mathsf{Enc}(pk, sk)$ is public. However, such an encryption C_{sk} can be

transformed into an encryption $\mathsf{Enc}(pk, f(sk))$ of an arbitrary function of sk thanks to the fully homomorphic properties of the FHE scheme. This leads to a conceptually very simple fully KDM-secure encryption scheme from polynomial assumptions (and ELFs). (We stress that we do not claim novelty for this observation. The connection between FHE and KDM security has already been observed in [BHHI10], and [DHRW16] have observed that the FHE construction of Canetti et al. preserves interesting properties of the underlying encryption scheme. However, [DHRW16] do not explicitly mention KDM security, and we find these consequences interesting enough to point out.)

As a second application, we consider spooky encryption (with CRS) introduced by Dodis et al. [DHRW16]. Note that this is not the contribution of this author and is hence not included in this thesis. We refer the reader to the original publication [ACH20] for more details.

We believe that our new notion of obfuscation will prove useful in other applications; for example, it would likely allow to improve the recent result [CRRV17], which constructed IND-CCA1-secure FHE from subexponentially secure IO.

FOLLOW-UP WORK. In the recent work [DN18], Döttling and Nishimaki define the notion universal proxy re-encryption (UPRE). UPRE schemes allow a proxy to convert any ciphertext under any public key of any existing PKE scheme into a ciphertext under any public key of any possibly different existing PKE scheme. [DN18] instantiate UPRE based on probabilistic IO due to [CLTV15]. UPRE for all PKE schemes (including non re-randomizable ones) requires dynamic-input pIO, which implies differing-inputs obfuscation. However, [DN18] observe that our notion of doubly-probabilistic IO suffices which yields an instantiation of UPRE for all PKE schemes based on polynomial IO and exponential DDH.

4 PRELIMINARIES

In this chapter, we introduce the necessary preliminaries for this part. In Section 4.1, we introduce a variant of pseudorandom functions and in Section 4.2 we introduce extremely lossy functions – the core tool for our construction in Chapter 6.

4.1 PERFECT PUNCTURABLE PRFS

In this part, we use puncturable PRFs with the additional property that a one-time evaluation is uniformly distributed over the output space.

DEFINITION 4.1 (Perfect pPRF, [DHRW16]). A *perfect* puncturable PRF is a family of puncturable PRFs F mapping $n(\lambda)$ bits to $m(\lambda)$ bits such that for all $\lambda \in \mathbb{N}$ and all inputs $x \in \{0,1\}^{n(\lambda)}$, the distribution

$$\{K \leftarrow \mathsf{F.KGen}(1^\lambda) \colon \mathsf{F.Eval}(K, x)\}$$

is identical to the uniform distribution over $\{0,1\}^{m(\lambda)}$.

Given any puncturable PRF F', we can build a perfect puncturable PRF F by sampling two keys $K_1 \leftarrow \mathsf{F}'.\mathsf{KGen}(1^\lambda)$ and $K_2 \leftarrow \{0,1\}^{m(\lambda)}$ in the key generation algorithm and defining the evaluation algorithm to output $\mathsf{F}'.\mathsf{Eval}(K_1, x) \oplus K_2$ on input of x, see [DHRW16].

4.2 EXTREMELY LOSSY FUNCTIONS

In this section, we present extremely lossy functions (ELFs) introduced in [Zha16]. Extremely lossy functions are a highly powerful primitive for complexity absorption allowing to replace subexponential or even exponential security assumptions with polynomial ones.

Informally, an ELF is a function that can be generated in two different modes, an injective mode and an extremely lossy mode. In injective mode, the range of the ELF has exponential size whereas the range comprises only polynomially many elements in extremely lossy mode. However, an adversary running in time $\mathcal{O}(\sqrt{r})$, where r denotes the cardinality of the range in extremely lossy mode, can distinguish the injective from the extremely lossy mode by evaluating the function on $\sqrt{r}$ inputs until he finds a collision. Hence, classic security against arbitrary polynomial time distinguishers between injective and lossy mode is impossible. However, security against all adversaries running in time at most r^c for $c < 1/2$ is still achievable. Consequently, the parameter r is chosen *depending* on the adversary $\mathcal{A}$, such that for any polynomial time adversary, there exists a polynomial r, such that $\mathcal{A}$ cannot distinguish between injective and lossy mode.

DEFINITION 4.2 (Extremely lossy function, [Zha16]). An *extremely lossy function (ELF)* is a tuple of PPT algorithms $\mathsf{ELF} = (\mathsf{Gen}, \mathsf{Eval})$ such that

- Gen on input of (M, r), where M is an integer and $r \in [M]$, outputs a function description $G\colon [M] \to [N]$.
- If $r = M$, G is injective with overwhelming probability (in $\log M$) over the randomness of $\mathsf{Gen}(M, M)$.
- For every $r \in [M]$, $|G([M])| < r$ with overwhelming probability (in $\log M$) over the randomness of $\mathsf{Gen}(M, r)$.
- $\mathsf{Eval}(G, \cdot)$ evaluates the function $G\colon [M] \to [N]$ and runs in polynomial time in $\log M$.

An extremely lossy function ELF is required to satisfy the following property.

INDISTINGUISHABILITY. For every large enough M, every polynomial P, and every inverse polynomial function δ, there exists a polynomial Q such that for every adversary $\mathcal{A}$ running in time at most $P(\log M)$ and every $r \in [Q(\log M), M]$, the advantage of $\mathcal{A}$

$$\begin{aligned}&\mathrm{Adv}^{\mathsf{elf}}_{\mathsf{ELF},(M,r),\mathcal{A}}(\log M) \coloneqq \\ &\quad \big|\Pr\left[G \leftarrow \mathsf{ELF.Gen}(M, M)\colon \mathcal{A}(\log M, G) = 1\right] \\ &\quad -\Pr\left[G \leftarrow \mathsf{ELF.Gen}(M, r)\colon \mathcal{A}(\log M, G) = 1\right]\big|\end{aligned}$$

is upper bounded by $\delta(\log M)$.

Note that the above definition does not explicitly use a security parameter λ. Instead, the bitlength of M (i. e., $\log M$) acts as security parameter. As a shorthand notation, we often write $G(x)$ to denote $\mathsf{ELF.Eval}(G, x)$. For our purpose, we consider extremely lossy functions satisfying additional properties.

DEFINITION 4.3 (Strong regularity, [Zha16]). An extremely lossy function $\mathsf{ELF} = (\mathsf{Gen}, \mathsf{Eval})$ is *strongly regular* if for every polynomial r, with overwhelming probability over the choice of $G \leftarrow \mathsf{Gen}(M, r)$, the distribution $\{x \leftarrow [M]\colon G(x)\}$ is statistically close to the uniform distribution over $G([M])$.

Note that if an ELF is strongly regular, it is possible to efficiently enumerate its image. The set of values obtained by evaluating an ELF on $\log(M) \cdot r \cdot \log(r)$ random inputs, where r is an upper bound on the size of its image, contains the entire image of the ELF with overwhelming probability.

4.2.1 Instantiating Extremely Lossy Functions

In [Zha16], Zhandry provides a construction of strongly regular extremely lossy functions based on the exponential hardness of the decisional Diffie-Hellman assumption eDDH (or any of its variants, such as the decision linear assumption). Let GGen be a group parameter generation algorithm. That is, GGen samples tuples of the form $(\mathbb{G}, p, g)$ such that $\mathbb{G}$ is a description of a cyclic group, p is its order and g is a generator of $\mathbb{G}$.

DEFINITION 4.4 (Exponential DDH assumption, [Zha16]). The *exponential decisional Diffie-Hellman (eDDH) assumption* is true relative to GGen if there exists a polynomial q such that the following is true. For every time bound

t and probability ϵ, let $\lambda \coloneqq \log q(t, 1/\epsilon)$. For every adversary $\mathcal{A}$ running in time at most t, the advantage

$$\left| \Pr\left[\begin{matrix} (\mathbb{G}, p, g) \leftarrow \mathsf{GGen}(1^\lambda) \\ (a, b, c) \leftarrow \mathbb{Z}_p^3 \end{matrix} : \mathcal{A}\left(\mathbb{G}, g, g^a, g^b, g^c\right) = 1 \right] \right.$$
$$\left. - \Pr\left[\begin{matrix} (\mathbb{G}, p, g) \leftarrow \mathsf{GGen}(1^\lambda) \\ (a, b) \leftarrow \mathbb{Z}_p^2 \end{matrix} : \mathcal{A}\left(\mathbb{G}, g, g^a, g^b, g^{ab}\right) = 1 \right] \right| \leqslant \epsilon.$$

As noted in [Zha16], groups based on elliptic curves are plausible candidates where this assumption holds. In practical instantiations of DDH over elliptic curves, the size of the group is chosen assuming that the best attack requires time in $\mathcal{O}(\sqrt{p})$, hence disproving eDDH (which amounts to showing that there is an attack which takes time less than p^c for any constant c) would have considerable practical implications. On the other hand, relying on some form of exponential hardness assumption seems necessary, since an instantiation from polynomial hardness would have surprising complexity-theoretic implications. More precisely, given access to only some super-logarithmic amount of non-determinism (i. e., $\omega(\log\log M)$ bits, where $[M]$ is the domain of the ELF), it is easy to distinguish between injective and lossy mode of the ELF. This is due to the fact that in lossy mode, the image of G has only polynomial size which means that the restriction of G to the set $D = [2^{\omega(\log\log M)}]$ (having super-polynomial cardinality) is guaranteed to have a collision (which is not the case in injective mode), and using only $\omega(\log\log M)$ bits of non-determinism this collision can be guessed.

NON-BLACK-BOX TECHNIQUES. Using extremely lossy function in a proof of security requires to explicitly use the size of the adversary as a circuit. Hence, the resulting proof does not treat the adversary as a black-box but is "slightly non-black-box". Consequently, in some cases, extremely lossy functions can be used to avoid black-box impossibility results.

5 DOUBLY-PROBABILISTIC IO

In this chapter, we define our novel variant of indistinguishability obfuscation for probabilistic circuits, which takes into account the fact that in many applications, obfuscated (probabilistic) circuits might only have to be evaluated on inputs coming from specific distributions.

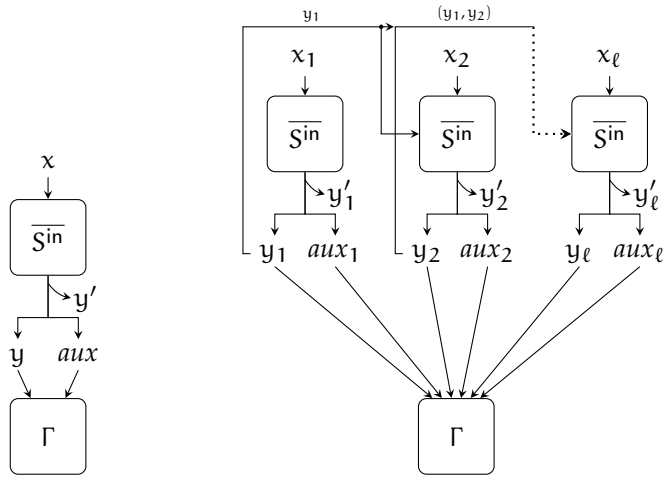
We leverage the fact that obfuscated circuits only have to work when fed with certain input distributions as follows. By "compiling" the input sampler S^{in} into a program which additionally produces auxiliary material *aux*, we introduce the possibility to verify that a given input was indeed produced according to S^{in} (on some input $m \in L$). We also refer to this auxiliary material as "certificate". As a result, an obfuscated circuit can always verify whether its input was sampled "honestly". If some input lacks a valid certificate, the obfuscated program simply outputs some error symbol $\perp$. In other words, we *restrict the correctness* of the obfuscated circuit to only hold for such well-formed inputs.

However, this approach faces two issues. First, the inputs to an obfuscated circuit might not be sampled "all at once" from a single distribution; rather, they can come from different and independent sources. We capture this behavior by defining *ℓ-source obfuscation*, to account for the fact that different inputs might have been sampled independently. Second, when inputs are sampled by different parties, there might still be interdependencies which must be accounted for. For example, a party might sample an input (e. g., a public key of an encryption scheme), pass it to a second party, who then samples a second input from a distribution that is parametrized by the first input (e. g., a ciphertext under that public key). We handle this possibility by ordering the ℓ inputs to the obfuscated circuit, and by considering a *stateful* input sampler S^{in}: when S^{in} is used to generate the i'th sample y_i, it receives in addition to its input a state $\mathsf{stf}(y_1, \ldots, y_{i-1})$, where stf is some fixed efficiently computable *state function* (which depends on the particular application), and the y_j (for $j \in [i-1]$) are outputs sampled by the first $i-1$ sources. The state function captures the fact that a particular application might define an arbitrary communication pattern, and specifies which samples a party should have access to when generating its sample.

In the following, we consider a circuit class $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda \in \mathbb{N}}$ of (probabilistic) circuits expecting ℓ inputs of length $n(\lambda)$, such that $\mathcal{C}_\lambda$ only contains circuits of size λ.

Additionally, we admit the possibility that a sampler produces some additional correlated output, that will not serve as input to an obfuscated circuit. Hence, there is no need to "certify" this input using the auxiliary information, and we call this output unauthenticated output. Continuing the use case from above, given a sampler producing some public key, the unauthenticated part of that sampler's output could be a corresponding secret key. Figure 5.1 visualizes the typical use case of a 1-source dpIO scheme (Figure 5.1a) and of an ℓ-source dpIO scheme (Figure 5.1b).

We now define input samplers more formally.



(a) A 1-source dpIO scheme. **(b)** An ℓ-source dpIO scheme.

Figure 5.1: Overview of our formalization for 1-source dpIO and ℓ-source dpIO. S^{in} denotes an input sampler, $\overline{S^{in}}$ denotes an encoded input sampler and Γ denotes the obfuscation of a circuit C with respect to the input sampler S^{in}. Figure 5.1a depicts a 1-source dpIO scheme. Hence, the input sampler is stateless. Figure 5.1b depicts an ℓ-source dpIO scheme. The state for the i-th sample of S^{in} may depend on all previous (authenticated) outputs $y_1, \ldots, y_{i-1}$. Formally, the state function derives the state from $y_1, \ldots, y_{i-1}$, but is omitted for simplicity.

DEFINITION 5.1 (Input sampler). An *input sampler* with input space L and state space $\mathcal{T}$ is a probabilistic algorithm S^{in} such that $S^{in}(\mathit{state}, x)$ produces outputs of the form $(y; y')$ with $y \in \{0,1\}^{n(|x|)}$ (for $\mathit{state} \in \mathcal{T}$ and $x \in L$). A class of input samplers $\mathcal{S}^{in}$ is an ensemble of input samplers sharing the same input space and state space.

We will treat $S^{in}(\cdot,\cdot)$ as a Boolean circuit. Looking ahead, the first part y will be used as input to the probabilistic circuit to be obfuscated, and hence needs to be authenticated. Since the second part y' is not meant as input to the probabilistic circuit, y' will not be authenticated. A state function stf efficiently maps $(\ell-1)$-tuples from $(\{0,1\}^{n(\lambda)} \cup \{\bot\})^{\ell-1}$ to states from $\mathcal{T}$.

We are now prepared to define our novel notion of probabilistic indistinguishability obfuscation. Since there are two probabilistic processes involved (the input sampling and the evaluation of the (obfuscated) probabilistic circuit), we call this notion doubly-probabilistic indistinguishability obfuscation.

DEFINITION 5.2 (Doubly-probabilistic indistinguishability obfuscation). Let ℓ be an integer. Let $\mathsf{stf}\colon (\{0,1\}^{n(\lambda)} \cup \{\bot\})^{\ell-1} \to \mathcal{T}$ be an efficiently computable state function. Let $\mathcal{S}^{in}$ be a class of input samplers with input space L and state space $\mathcal{T}$. Let $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda\in\mathbb{N}}$ be a family of (probabilistic) circuits expecting ℓ inputs of length $n(\lambda)$, and let $\mathfrak{C}$ be a class of circuit samplers over $\mathcal{C}$. An *ℓ-source doubly-probabilistic indistinguishability obfuscation (dpIO) scheme* for $(\mathsf{stf}, \mathcal{S}^{in}, \mathcal{C}, \mathfrak{C})$ is a triple of PPT algorithms $\mathsf{dpiO} = (\mathsf{Setup}, \mathsf{Encode}, \mathsf{Obfuscate})$ such that

- $\mathsf{Setup}(1^\lambda)$ outputs public parameters pp,

- $\mathsf{Encode}(pp, \mathsf{S}^{\mathsf{in}})$, on input of pp and an input sampler $\mathsf{S}^{\mathsf{in}} \in \mathcal{S}^{\mathsf{in}}$, the *deterministic* algorithm Encode outputs an encoded input sampler $\overline{\mathsf{S}^{\mathsf{in}}}$,
- $\mathsf{Obfuscate}(pp, \mathsf{S}^{\mathsf{in}}, \mathsf{C})$, on input of pp, an input sampler $\mathsf{S}^{\mathsf{in}} \in \mathcal{S}^{\mathsf{in}}$ and a circuit $\mathsf{C} \in \mathcal{C}_{\ell \cdot \lambda}$, Obfuscate outputs a circuit Γ of size $poly(\lambda, |\mathsf{C}|)$. We call Γ an *obfuscation of* C *with respect to* S^{in}.

A dpIO scheme is required to satisfy *simulatability of encodings* (Definition 5.3), *restricted correctness* (Definition 5.4) and *security with respect to the class of circuit samplers* $\mathfrak{C}$ (Definition 5.5).

Given the definition above, some remarks are in order. A dpIO scheme is parametrized not only by a circuit class $\mathcal{C}$ and a circuit sampler class $\mathfrak{C}$, but also by an input sampler class $\mathcal{S}^{\mathsf{in}}$ and a state function stf. Circuits are always obfuscated with respect to some input sampler S^{in}. As a consequence, the obfuscated circuit expects inputs from the corresponding encoded input sampler $\overline{\mathsf{S}^{\mathsf{in}}}$. Furthermore, the obfuscation procedure takes circuits $\mathsf{C} \in \mathcal{C}_\lambda$ as input. That is, C expects ℓ inputs of length $n(\lambda)$ and $|\mathsf{C}| = \lambda$.

Informally, simulatability of encodings ensures that, on any (adversarially chosen) input x, state *state*, and input sampler S^{in}, the encoded sampler $\overline{\mathsf{S}^{\mathsf{in}}}$, on input of $(state, x)$, outputs samples of the form $(y, aux; y')$, where $(y; y')$ is distributed as outputs from $\mathsf{S}^{\mathsf{in}}(state, x)$, and *aux* does not leak any nontrivial information about the inputs. This is formalized by requiring the existence of a simulator that can simulate *aux* given only y.

$\mathrm{Exp}^{(0)\text{-enc}}_{\mathsf{dpiO},\mathcal{A}}(\lambda)$

$pp \leftarrow \mathsf{Setup}(1^\lambda)$

return $\mathcal{A}^{\mathcal{O}^{(0)\text{-enc}}(\cdot,\cdot,\cdot)}(pp)$

(a) Game for the simulatability of encodings property, where $\mathcal{A}$ is given honestly generated public parameters and triplets $(y, aux; y')$ produced by the encoded sampler.

$\mathrm{Exp}^{(1)\text{-enc}}_{\mathsf{Sim},\mathcal{A}}(\lambda)$

$(pp, td) \leftarrow \mathsf{Sim}_0(1^\lambda)$

return $\mathcal{A}^{\mathcal{O}^{(1)\text{-enc}}(\cdot,\cdot,\cdot)}(pp)$

(b) Game for the simulatability of encodings property, where $\mathcal{A}$ is fed with simulated public parameters and simulated triplets $(y, aux; y')$.

Oracle $\mathcal{O}^{(0)\text{-enc}}(\mathsf{S}^{\mathsf{in}}, state, x)$

$\overline{\mathsf{S}^{\mathsf{in}}} \leftarrow \mathsf{Encode}(pp, \mathsf{S}^{\mathsf{in}})$

$(y, aux; y') \leftarrow \overline{\mathsf{S}^{\mathsf{in}}}(state, x)$

return $(y, aux; y')$

(c) Oracle producing triplets $(y, aux; y')$ using the encoded sampler.

Oracle $\mathcal{O}^{(1)\text{-enc}}(\mathsf{S}^{\mathsf{in}}, state, x)$

$(y; y') \leftarrow \mathsf{S}^{\mathsf{in}}(state, x)$

$aux \leftarrow \mathsf{Sim}_1(pp, td, \mathsf{S}^{\mathsf{in}}, y, state)$

return $(y, aux; y')$

(d) Oracle simulating triplets $(y, aux; y')$ using the original input sampler.

Figure 5.2: Definitions of the security games for the simulatability of encodings property of dpIO schemes. The PPT algorithm $\mathcal{A}$ can interact polynomially many times with either $\mathcal{O}^{(0)\text{-enc}}$ (Figure 5.2c) or $\mathcal{O}^{(1)\text{-enc}}$ (Figure 5.2d). Note that $\mathcal{A}$ provides the input sampler S^{in} as a Boolean circuit.

DEFINITION 5.3 (Simulatability of encodings). An ℓ-source dpIO scheme dpiO for $(\mathsf{stf}, \mathcal{S}^{\mathsf{in}}, \mathcal{C}, \mathfrak{C})$ satisfies *simulatability of encodings* if for every PPT adversary $\mathcal{A}$, there exists a PPT simulator $\mathsf{Sim} = (\mathsf{Sim}_0, \mathsf{Sim}_1)$ such that

$$\mathrm{Adv}^{\mathsf{enc}}_{\mathsf{dpiO},\mathsf{Sim},\mathcal{A}}(\lambda) := \left| \Pr\left[\mathrm{Exp}^{(0)\text{-enc}}_{\mathsf{dpiO},\mathcal{A}}(\lambda) = 1\right] - \Pr\left[\mathrm{Exp}^{(1)\text{-enc}}_{\mathsf{Sim},\mathcal{A}}(\lambda) = 1\right] \right|$$

is negligible, where the games $\mathrm{Exp}^{(0)\text{-enc}}_{\mathsf{dpiO},\mathcal{A}}(\lambda)$ and $\mathrm{Exp}^{(1)\text{-enc}}_{\mathsf{Sim},\mathcal{A}}(\lambda)$ are defined in Figure 5.2.

Intuitively, restricted correctness ensures that in an "honest scenario", i. e., when the inputs $(y_1, \ldots, y_\ell)$ are sampled using the input sampler S^{in}, then the obfuscated circuit is guaranteed to behave "correctly". More precisely, restricted correctness guarantees that the support of the original circuit is respected, that the outputs of the obfuscated (deterministic) circuit are computationally indistinguishable from once-per-input evaluations of the original (probabilistic) circuit, and that the output distribution of the obfuscated circuit (taken over the random coins of the obfuscator) is one-time statistically close to the output distribution of the original circuit (taken over its internal random coins). Note that the statistical indistinguishability does not extend to multiple evaluations.

To make this definition meaningful, we need a way to let the obfuscated circuit verify that the inputs are well-formed. We do not require that the inputs were generated through S^{in} with uniformly random coins, but only that they were generated through S^{in} with *some* random coins (and some input). To enable verification, we let the parties generate their input using the encoded sampler $\overline{S^{\mathsf{in}}}$ instead. An encoded sampler produces inputs as S^{in}, and in addition produces auxiliary information which can be used by the obfuscated program to verify that the inputs were honestly constructed. More formally, for a given input y, the obfuscated circuit verifies that there exists an input x, random coins r, and an unauthenticated part y' such that $(y; y') = S^{\mathsf{in}}(x; r)$.

$\mathrm{Exp}^{(0)\text{-close}}_{\mathsf{dpiO},S^{\mathsf{in}},(x_1,\ldots,x_\ell),C,\mathcal{A}}(\lambda)$

$pp \leftarrow \mathsf{Setup}(1^\lambda)$

$\overline{S^{\mathsf{in}}} \leftarrow \mathsf{Encode}(pp, S^{\mathsf{in}})$

$\Gamma \leftarrow \mathsf{Obfuscate}(pp, S^{\mathsf{in}}, C)$

for $j \in [\ell]$ **do**

$\quad state_j \leftarrow \mathsf{stf}(y_1, \ldots, y_{j-1}, \bot, \ldots)$

$\quad (y_j, aux_j; y'_j) \leftarrow \overline{S^{\mathsf{in}}}(state_j, x_j)$

$z \leftarrow \Gamma(y_1, aux_1, \ldots, y_\ell, aux_\ell)$

return $\mathcal{A}(pp, C, S^{\mathsf{in}}, (x_i, y_i)_{i\in[\ell]}, z)$

(a) Closeness game, where $\mathcal{A}$ receives a single output from the obfuscated circuit.

$\mathrm{Exp}^{(1)\text{-close}}_{\mathsf{dpiO},S^{\mathsf{in}},(x_1,\ldots,x_\ell),C,\mathcal{A}}(\lambda)$

$pp \leftarrow \mathsf{Setup}(1^\lambda)$

for $j \in [\ell]$ **do**

$\quad state_j \leftarrow \mathsf{stf}(y_1, \ldots, y_{j-1}, \bot, \ldots)$

$\quad (y_j; y'_j) \leftarrow S^{\mathsf{in}}(state_j, x_j)$

$z \leftarrow C(y_1, \ldots, y_\ell)$

return $\mathcal{A}(pp, C, S^{\mathsf{in}}, (x_i, y_i)_{i\in[\ell]}, z)$

(b) Closeness game, where $\mathcal{A}$ receives a single output from the original randomized circuit.

Figure 5.3: Games $\mathrm{Exp}^{(b)\text{-close}}_{\mathsf{dpiO},S^{\mathsf{in}},(x_1,\ldots,x_\ell),C,\mathcal{A}}(\lambda)$ (for $b \in \{0,1\}$) modeling the statistical closeness property of an ℓ-source dpIO scheme.

A small technicality is that we must allow the input sampler to depend on state information, to capture interdependencies between the inputs. This means that the auxiliary information will have to certify that an input was generated correctly, with respect to some state that the obfuscated circuit might not have access to (which would prevent it from verifying the certificate). However, this issue disappears by restricting the interdependencies to only involve a state computed from the *previous samples* (as opposed to more

complex interdependencies which would involve, for example, the coins used to produce these samples). See also Figure 5.1b for an overview. In this case, the obfuscated circuit can test the certificates in an incremental way. The first test verifies that y_1 was correctly constructed with respect to the state $\mathsf{stf}(\bot, \ldots, \bot)$, the second test verifies that y_2 was correctly constructed with respect to the state $\mathsf{stf}(y_1, \bot, \ldots, \bot)$, and the i-th test verifies that y_i was correctly constructed with respect to the state $\mathsf{stf}(y_1, \ldots, y_{i-1}, \bot, \ldots, \bot)$.

DEFINITION 5.4 (Restricted correctness). An ℓ-source dpIO scheme dpiO for $(\mathsf{stf}, \mathcal{S}^{\mathsf{in}}, \mathcal{C}, \mathfrak{C})$ satisfies *restricted correctness* if the following properties are satisfied.

STATISTICAL CLOSENESS. One-time evaluation of the encoded input sampler and the obfuscated circuit (on these inputs) is statistically close to one-time evaluation of the original input sampler and the original circuit (on these inputs). That is, for every $S^{\mathsf{in}} \in \mathcal{S}^{\mathsf{in}}$, $(x_1, \ldots, x_\ell) \in L^\ell$, and $C \in \mathcal{C}_\lambda$, for every (possibly unbounded) adversary $\mathcal{A}$, the advantage

$$\mathrm{Adv}^{\mathsf{close}}_{\mathsf{dpiO}, S^{\mathsf{in}}, (x_1, \ldots, x_\ell), C, \mathcal{A}}(\lambda) \coloneqq \left| \Pr\left[\mathrm{Exp}^{(0)\text{-}\mathsf{close}}_{\mathsf{dpiO}, S^{\mathsf{in}}, (x_1, \ldots, x_\ell), C, \mathcal{A}}(\lambda) = 1\right] - \Pr\left[\mathrm{Exp}^{(1)\text{-}\mathsf{close}}_{\mathsf{dpiO}, S^{\mathsf{in}}, (x_1, \ldots, x_\ell), C, \mathcal{A}}(\lambda) = 1\right] \right|$$

is negligible, where $\mathrm{Exp}^{(b)\text{-}\mathsf{close}}_{\mathsf{dpiO}, S^{\mathsf{in}}, (x_1, \ldots, x_\ell), C, \mathcal{A}}(\lambda)$ (for $b \in \{0, 1\}$) is defined in Figure 5.3.

COMPUTATIONAL CORRECTNESS. Evaluations of an obfuscated program are computationally indistinguishable from evaluations of the original randomized program on well-formed inputs. That is, for every $S^{\mathsf{in}} \in \mathcal{S}^{\mathsf{in}}$ and $C \in \mathcal{C}_\lambda$, for every PPT adversary $\mathcal{A}$, the advantage

$$\mathrm{Adv}^{\mathsf{rcorr}}_{\mathsf{dpiO}, C, \mathcal{A}}(\lambda) \coloneqq \left| \Pr\left[\mathrm{Exp}^{(0)\text{-}\mathsf{rcorr}}_{\mathsf{dpiO}, C, \mathcal{A}}(\lambda) = 1\right] - \Pr\left[\mathrm{Exp}^{(1)\text{-}\mathsf{rcorr}}_{\mathsf{dpiO}, C, \mathcal{A}}(\lambda) = 1\right] \right|$$

is negligible, where $\mathrm{Exp}^{(b)\text{-}\mathsf{rcorr}}_{\mathsf{dpiO}, C, \mathcal{A}}(\lambda)$ (for $b \in \{0, 1\}$) is defined in Figure 5.4 and the oracles $\mathcal{O}^{(0)\text{-}\mathsf{rcorr}}$ and $\mathcal{O}^{(1)\text{-}\mathsf{rcorr}}$ must not be called twice on the same input.

SUPPORT RESPECTING. The encoded sampler $\overline{S^{\mathsf{in}}}$ and the obfuscated circuit Γ respect the support of the original sampler S^{in} and the original circuit C, respectively. That is, for all $pp \in \mathit{supp}(\mathsf{Setup}(1^\lambda))$, $\overline{S^{\mathsf{in}}} \coloneqq \mathsf{Encode}(pp, S^{\mathsf{in}})$ and all $\Gamma \in \mathit{supp}(\mathsf{Obfuscate}(pp, S^{\mathsf{in}}, C))$, we have that

- for all $x \in L$, $\mathit{state} \in \mathcal{T}$, $(y, \mathit{aux}; \cdot) \in \mathit{supp}(\overline{S^{\mathsf{in}}}(\mathit{state}, x))$, we have that $(y; \cdot) \in \mathit{supp}(S^{\mathsf{in}}(\mathit{state}, x))$, and
- for all inputs $x_1, \ldots, x_\ell \in L$, all $j \in [\ell]$, all inputs $(y_j, \mathit{aux}_j; \cdot) \in \mathit{supp}(\overline{S^{\mathsf{in}}}(\mathit{state}_j, x_j))$, where $\mathit{state}_j \coloneqq \mathsf{stf}(y_1, \ldots, y_{j-1}, \bot, \ldots, \bot)$, we have that

$$\Gamma(y_1, \mathit{aux}_1, \ldots, y_\ell, \mathit{aux}_\ell) \in \mathit{supp}(C(y_1, \ldots, y_\ell)).$$

Note that closeness implies that samples produced with the original input sampler S^{in} and samples produced with the encoded input sampler $\overline{S^{\mathsf{in}}}$ are statistically close.

$\mathrm{Exp}^{(0)\text{-rcorr}}_{\mathsf{dpiO},C,\mathcal{A}}(\lambda)$

$pp \leftarrow \mathsf{Setup}(1^\lambda)$

$\overline{S^{\mathsf{in}}} \leftarrow \mathsf{Encode}(pp, S^{\mathsf{in}})$

$\Gamma \leftarrow \mathsf{Obfuscate}(pp, S^{\mathsf{in}}, C)$

return $\mathcal{A}^{\mathcal{O}^{(0)\text{-rcorr}}}(1^\lambda, pp, C)$

(a) Correctness game, where $\mathcal{A}$ has oracle access to obfuscated circuit.

$\mathrm{Exp}^{(1)\text{-rcorr}}_{\mathsf{dpiO},C,\mathcal{A}}(\lambda)$

return $\mathcal{A}^{\mathcal{O}^{(1)\text{-rcorr}}}(1^\lambda, pp, C)$

(b) Correctness game, where $\mathcal{A}$ has oracle access to original randomized circuit.

$\mathcal{O}^{(0)\text{-rcorr}}(x_1, \ldots, x_\ell)$

for $j \in [\ell]$ **do**

$state_j \leftarrow \mathsf{stf}(y_1, \ldots, y_{j-1}, \bot, \ldots)$

$(y_j, aux_j; y'_j) \leftarrow \overline{S^{\mathsf{in}}}(state_j, x_j)$

return $\Gamma(y_1, aux_1, \ldots, y_\ell, aux_\ell)$

(c) Oracle which evaluates the obfuscated circuit.

$\mathcal{O}^{(1)\text{-rcorr}}(x_1, \ldots, x_\ell)$

for $j \in [\ell]$ **do**

$state_j \leftarrow \mathsf{stf}(y_1, \ldots, y_{j-1}, \bot, \ldots)$

$(y_j; y'_j) \leftarrow S^{\mathsf{in}}(state_j, x_j)$

return $C(y_1, \ldots, y_\ell)$

(d) Oracle which evaluates the original randomized circuit.

Figure 5.4: Experiments $\mathrm{Exp}^{(0)\text{-rcorr}}_{\mathsf{dpiO},C,\mathcal{A}}(\lambda)$ and $\mathrm{Exp}^{(1)\text{-rcorr}}_{\mathsf{dpiO},C,\mathcal{A}}(\lambda)$ modeling the computational correctness property an ℓ-source dpIO scheme.

Our notion of security is close in spirit to the standard indistinguishability notion for obfuscation of probabilistic circuits of [CLTV15] (see Definition 2.16). However, in our scenario, the security notion must account for the fact that public parameters pp are generated in a setup phase; the indistinguishability property of obfuscated circuits must therefore hold when many circuits are obfuscated with respect to the *same* public parameters. This suggests an oracle-based security notion.

$\mathrm{Exp}^{(b)\text{-ind}}_{\mathsf{dpiO},D,\mathcal{A}}(\lambda)$

$pp \leftarrow \mathsf{Setup}(1^\lambda)$

return $\mathcal{A}^{\mathcal{O}^{(b)\text{-ind}}(\cdot)}(pp)$

(a) Game $\mathrm{Exp}^{(b)\text{-ind}}_{\mathsf{dpiO},D,\mathcal{A}}(\lambda)$.

Oracle $\mathcal{O}^{(b)\text{-ind}}(S^{\mathsf{in}})$

$(C_0, C_1, z) \leftarrow D_\lambda$

$\Gamma \leftarrow \mathsf{Obfuscate}(pp, S^{\mathsf{in}}, C_b)$

return (C_0, C_1, z, Γ)

(b) Oracle $\mathcal{O}^{(b)\text{-ind}}$ producing obfuscations for circuit C_b.

Figure 5.5: Definition of the games $\mathrm{Exp}^{(b)\text{-ind}}_{\mathsf{dpiO},D,\mathcal{A}}(\lambda)$ (for $b \in \{0,1\}$) modeling the indistinguishability property of a dpIO scheme. The adversary $\mathcal{A}$ can interact polynomially many times with the oracle $\mathcal{O}^{(b)\text{-ind}}$. Given a description of an input sampler S^{in}, the oracle $\mathcal{O}^{(b)\text{-ind}}$ samples (C_0, C_1, z) from the circuit sampler and obfuscates C_b with respect to S^{in}.

DEFINITION 5.5 (Security with respect to $\mathfrak{C}$). An ℓ-source dpIO scheme dpiO for $(\mathsf{stf}, \mathcal{S}^{\mathsf{in}}, \mathcal{C}, \mathfrak{C})$ satisfies *security with respect to* $\mathfrak{C}$ if for every circuit sampler $D = \{D_\lambda\}_{\lambda \in \mathbb{N}} \in \mathfrak{C}$, for every PPT adversary $\mathcal{A}$, the advantage

$$\mathrm{Adv}^{\mathsf{ind}}_{\mathsf{dpiO},D,\mathcal{A}}(\lambda) := \left| \Pr\left[\mathrm{Exp}^{(0)\text{-ind}}_{\mathsf{dpiO},D,\mathcal{A}}(\lambda) = 1\right] - \Pr\left[\mathrm{Exp}^{(1)\text{-ind}}_{\mathsf{dpiO},D,\mathcal{A}}(\lambda) = 1\right] \right|$$

is negligible, where the games $\mathrm{Exp}^{(0)\text{-ind}}_{\mathsf{dpiO},\mathsf{D},\mathcal{A}}(\lambda)$ and $\mathrm{Exp}^{(1)\text{-ind}}_{\mathsf{dpiO},\mathsf{D},\mathcal{A}}(\lambda)$ are defined in Figure 5.5.

6 CONSTRUCTION

Before describing our construction, we recall some notation for this chapter. Let $\ell \in \mathbb{N}$ be some constant. Let $\mathcal{C} = (\mathcal{C}_\lambda)_{\lambda\in\mathbb{N}}$ be a circuit class such that $\mathcal{C}_\lambda$ contains (arbitrary) circuits of size λ. Recall that $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda\in\mathbb{N}}$ is a class of (probabilistic) circuits expecting ℓ inputs of length $n(\lambda)$, such that $\mathcal{C}_\lambda$ only contains circuits of size λ.

Let $\mathcal{S}^{\mathsf{in}}$ be a class of (arbitrary) input samplers with *polynomially sized input space* L, that is $|L \cap \{0,1\}^\lambda| = \mathit{poly}(\lambda)$ for some polynomial $\mathit{poly}(\lambda)$. Note that such input samplers can produce exponentially many y and, in fact, these are the cases of interest for our applications.

In this chapter, we construct an ℓ-source dpIO scheme, for the input sampler class $\mathcal{S}^{\mathsf{in}}$, and dynamic-input indistinguishable circuit samplers for $\mathcal{C}$. Our construction relies on polynomially secure indistinguishability obfuscation (Definition 2.15), polynomially secure (perfect) puncturable pseudorandom functions (Definition 4.1), statistically sound non-interactive zero-knowledge proof systems (Definition 2.25), and an extremely lossy function (Definition 4.2).

6.1 OVERVIEW

We start by providing a high-level overview of our construction. The setup algorithm Setup generates parameters G for the extremely lossy function (in injective mode) and a common reference string σ for the non-interactive zero-knowledge proof system.

Given an input sampler $S^{\mathsf{in}} \in \mathcal{S}^{\mathsf{in}}$, the encode algorithm Encode compiles S^{in} to an encoded input sampler $\overline{S^{\mathsf{in}}}$ as follows. On input of $(\mathit{state}, x; r)$, $\overline{S^{\mathsf{in}}}$ samples $(y; y')$ using the input sampler $S^{\mathsf{in}}(\mathit{state}, x; G(r))$, certifies y with a NIZK proof $\pi =: \mathit{aux}$ produced via $\mathsf{NIZK.Prove}(\sigma, y, L^{G,S^{\mathsf{in}}}_{\mathit{state}}, (y', x, r))$, and outputs $(y, \mathit{aux}; y')$. The language $L^{G,S^{\mathsf{in}}}_{\mathit{state}}$ contains all values y for which there exists (y', x, r) such that $(y; y') = S^{\mathsf{in}}(\mathit{state}, x, G(r))$. We refer to values $y \in L^{G,S^{\mathsf{in}}}_{\mathit{state}}$ as well-formed inputs. Note that if G is in injective mode, $L^{G,S^{\mathsf{in}}}_{\mathit{state}}$ is, in general, a trivial language, i. e., the union over all $x \in L$ of the first component of $\mathit{supp}(S^{\mathsf{in}}(\mathit{state}, x))$. Informally speaking, simulatability of the encodings directly follows from the injectivity of G, and the zero-knowledge property of the proof system.

For the sake of simplicity, we describe the intuition behind our construction for a 1-source dpIO scheme. The obfuscator Obfuscate, on input of an input sampler S^{in} and a circuit C, obfuscates C (with respect to S^{in}) as follows. First, Obfuscate samples a PRF key K for F. Then, it constructs a Boolean circuit $\overline{C}$ which, on input of (y, aux), verifies the certificate aux with NIZK.Verify. If the test passes, $\overline{C}$ evaluates the circuit C on input of y and random coins $F(K, y)$. If the test fails, $\overline{C}$ outputs an error symbol $\perp$. Informally, restricted correctness follows from the correctness properties of all used components.

Recall that our main concern in this part is to avoid a subexponential reduction to indistinguishability obfuscation. Hence, the tricky part is to achieve security with respect to a preferably large class $\mathfrak{C}$ of circuit samplers. Interestingly, we are able to prove our dpIO scheme secure with respect to the class of dynamic-input indistinguishable samplers $\mathfrak{C}^{\text{dyn-ind}}$ over the class of all polynomial sized circuits. This even exceeds the capabilities of the pIO construction due to [CLTV15], since general pIO for the class of dynamic-input indistinguishable samplers falls victim to the implausibility result due to [GGHW17]. We avoid this implausibility by restricting the correctness of obfuscated programs to only hold for "certified inputs".

Therefore, to prove indistinguishability between the obfuscations of two dynamic-input indistinguishable circuits (C_0, C_1), we resort to the standard puncturing strategy of [CLTV15]. That is, we proceed through a sequence of hybrids, successively modifying of the obfuscated circuit, an input at a time. More precisely, starting with an obfuscation of C_0, we program the obfuscated circuit to behave like $C_1(y;r)$ instead of $C_0(y;r)$, for each input y. Using this naive approach directly, each such replacement relies on the security of the underlying IO scheme (and the security of the PRF, and the dynamic-input indistinguishability of C_0 and C_1).

The main issue of this approach is that the number of possible inputs y (hence the number of hybrids) is exponential – indeed, this is the reason why the piO scheme of [CLTV15] requires subexponentially secure primitives (iO and F). To get around this issue, we leverage the origin of inputs to circuits in conjunction with the power of the extremely lossy function G. We first switch G to an appropriate extremely lossy mode, which by definition cannot be distinguished from the injective mode. Now, the soundness of the NIZK proof system ensures that all valid inputs y originate from $S^{\text{in}}(\mathit{state}, x; G(r))$ for some (x, r) (omitting y' for simplicity). For a given *state*, the quantity of such values is bounded by the size of the range of G – which is merely polynomial – times the size of the input domain (of the input sampler) $L \cap \{0,1\}^\lambda$. Therefore, in all applications where the inputs to the obfuscated circuit are sampled using sampler inputs from a small domain, we can base security on *polynomially secure* iO. Note that for $\ell > 1$, the state input *state* for the input sampler S^{in} depends in a deterministic manner (the state function) from previous samples, which exist only in polynomial quantity. Therefore, the above approach generalizes to ℓ-source dpIO.

6.2 CONSTRUCTING DOUBLY-PROBABILISTIC IO

For our construction, we employ a perfectly sound NIZK proof system for the following (parametrized) language

$$L_{\mathit{state}}^{G,S^{\text{in}}} := \left\{ y \;\middle|\; \begin{array}{l} \exists\ y' \in \{0,1\}^* \\ \exists\ x \in L \cap \{0,1\}^\lambda : (y;y') = S^{\text{in}}(\mathit{state}, x; G(r)) \\ \exists\ r \in \{0,1\}^{k(\lambda)} \end{array} \right\}.$$

Note that $L_{\mathit{state}}^{G,S^{\text{in}}}$ is well-defined for arbitrary values of *state*, i. e., even for state information which cannot occur during honest use. Looking ahead,

validity of the state information is ensured independently by the obfuscated circuit.

For simplicity in the analysis, we use a NIZK proof system that satisfies the following property. With overwhelming probability over the coins of $\mathsf{Setup}(1^\lambda)$, there does not exist any pair (x, π) such that $x \notin L$ and $\mathsf{NIZK.Verify}(\sigma, x, \pi) = 1$. We call a NIZK that satisfies this property *almost perfectly sound*. We note that there is a simple folklore method which allows to construct an almost perfectly sound NIZK proof system starting from any statistically sound NIZK proof system. Consider a $2^{-\lambda}$-statistically sound NIZK proof system, for statements $x \in \{0,1\}^n$, for some polynomial $n = n(\lambda)$. Using parallel repetitions, the soundness of the proof system can be amplified to $2^{-\lambda-n}$. That is, for any statement $x \notin L$, the probability

$$\Pr_{\sigma}[\exists \pi\colon \mathsf{Verify}(\sigma, x, \pi) = 1] \leqslant 2^{-\lambda-n}.$$

Then, it necessarily holds that for all possible σ except a $2^{-\lambda}$ fraction of them, there does not exist any pair (x, π) such that $x \notin L$ and π is an accepting proof. To realize this, let E_x^σ denote the event that there exists a proof π such that $\mathsf{Verify}(\sigma, x, \pi) = 1$. Then, by a union bound argument,

$$\Pr_{\sigma}[\exists x \in \{0,1\}^n \setminus L\colon E_x^\sigma] \leqslant \sum_{x \in \{0,1\}^n \setminus L} \Pr_{\sigma}[E_x^\sigma] \leqslant 2^n \cdot 2^{-\lambda-n}.$$

Hence, the NIZK proof system obtained via parallel repetitions is almost perfectly sound.

Let $\ell \in \mathbb{N}$ be a constant, let $\mathsf{stf}\colon (\{0,1\}^{n(\lambda)} \cup \{\bot\})^{\ell-1} \to \mathcal{T}$ be an efficiently computable state function, and let $\mathcal{C} = \{\mathcal{C}_\lambda\}_\lambda$ be a family of (randomized) circuits taking ℓ inputs of size $n(\lambda)$ such that $\mathcal{C}_\lambda$ only contains circuits of size λ. Let $\mathcal{S}^{\mathsf{in}}$ be a class of (arbitrary) input samplers with randomness space $\{0,1\}^{k(\lambda)}$ polynomially sized input space L, that is $|L \cap \{0,1\}^\lambda| = \mathit{poly}(\lambda)$ for some polynomial $\mathit{poly}(\lambda)$. Further, let $\mathfrak{C}^{\text{dyn-ind}}$ be the class of dynamic-input indistinguishable samplers (over $\mathcal{C}$).

THEOREM 6.1. *Let* ELF *be a strongly regular extremely lossy function, let* iO *be a perfectly correct polynomially secure indistinguishability obfuscator, let* F *be a polynomially secure (perfect) puncturable PRF, and let* NIZK *be a statistically sound and computationally zero-knowledge NIZK proof system for the family of languages* $\{L_{\mathit{state}}^{G,\mathcal{S}^{\mathsf{in}}}\}_{\mathit{state},G,\mathcal{S}^{\mathsf{in}}}$. *Then, the dpIO scheme* $\mathsf{dpiO} = (\mathsf{Setup}, \mathsf{Encode}, \mathsf{Obfuscate})$ *defined in Figure 6.1 is an ℓ-source dpIO scheme for* $(\mathsf{stf}, \mathcal{S}^{\mathsf{in}}, \mathcal{C}, \mathfrak{C}^{\text{dyn-ind}})$.

Statistically sound perfectly correct NIZK proof systems for $\mathcal{NP}$ can be constructed from polynomially secure indistinguishability obfuscation and extremely lossy functions [BP15a]. Further, ELFs imply the existence of one-way functions, hence of (perfect) puncturable PRFs [GGM84a; HILL99]. Together with Theorem 6.1, we obtain the following corollary.

COROLLARY 6.1. *Assuming polynomially secure indistinguishability obfuscation and extremely lossy functions, for every constant ℓ, for every class of input samplers $\mathcal{S}^{\mathsf{in}}$ with a polynomial sized input domain, for every state function* stf *and for every probabilistic circuit class $\mathcal{C}$, there exists an ℓ-source doubly-probabilistic indistinguishability obfuscation scheme for* $(\mathsf{stf}, \mathcal{S}^{\mathsf{in}}, \mathcal{C}, \mathfrak{C}^{\text{dyn-ind}})$, *where $\mathfrak{C}^{\text{dyn-ind}}$ is the class of dynamic-input circuit samplers for $\mathcal{C}$.*

Proof of Theorem 6.1. We prove that dpiO as defined in Figure 6.1 satisfies simulatability of encodings (cf. Definition 5.3), statistical restricted correctness (cf. Definition 5.4), and indistinguishability (cf. Definition 5.5).

Setup(1^λ)

$\sigma \leftarrow \mathsf{NIZK.Setup}(1^\lambda)$

$G \leftarrow \mathsf{ELF.Gen}(2^{k(\lambda)}, 2^{k(\lambda)})$

$pp := (\sigma, G)$

return pp

(a) Setup algorithm producing public parameters.

Encode(pp, S^{in})

define $\overline{S^{in}}$ as follows :

Circuit $\overline{S^{in}}[pp](state, x; r_1, r_2)$

$(y; y') \leftarrow S^{in}(state, x; G(r_1))$

$\pi \leftarrow \mathsf{NIZK.Prove}(\sigma,$
$\quad st = (G, S^{in}, state, y),$
$\quad w = (y', x, r_1); r_2)$

return $(y, \pi; y')$

return $\overline{S^{in}}$

(b) Encoding algorithm which compiles input samplers S^{in} to so-called encoded input samplers $\overline{S^{in}}$.

Obfuscate(pp, S^{in}, C)

parse $pp =: (\sigma, G)$

$K \leftarrow \mathsf{F.KGen}(1^\lambda)$

define $\overline{C}$ as follows:

Circuit $\overline{C}[\mathsf{stf}, (\sigma, G), S^{in}, C, K](\nu)$

parse $\nu =: ((y_1, aux_1), \ldots, (y_\ell, aux_\ell))$

$state_j := \mathsf{stf}(y_1, \ldots, y_{j-1}, \bot, \ldots, \bot)$ for $i \in [\ell]$

if $\neg\big(\forall j \in [\ell]\colon \mathsf{NIZK.Verify}\big(\sigma, (G, S^{in}, state_j, y_j), aux_j\big) = 1\big)$ **then**

return $\bot$

$r := \mathsf{F.Eval}\big(K, (y_1, \ldots, y_\ell)\big)$

$\mu := C\big((y_1, \ldots, y_\ell); r\big)$

return μ

$\Lambda \leftarrow \mathsf{iO}(\overline{C})$

return $\Gamma := \Lambda$

(c) Obfuscate algorithm producing obfuscations Γ of circuits C with respect to an input sampler S^{in}.

Figure 6.1: Our construction of an ℓ-source dpIO scheme $\mathsf{dpiO} = (\mathsf{Setup}, \mathsf{Encode}, \mathsf{Obfuscate})$.

SIMULATABILITY OF ENCODINGS. We prove that there exists a PPT simulator $\mathsf{Sim} = (\mathsf{Sim}_0, \mathsf{Sim}_1)$ such that for every PPT adversary $\mathcal{A}$, the advantage $\mathrm{Adv}^{\mathsf{enc}}_{\mathsf{dpiO},\mathsf{Sim},\mathcal{A}}(\lambda)$ is negligible. By the zero-knowledge property of NIZK, there exists a simulator $\mathsf{NIZK.Sim} = (\mathsf{NIZK.Sim}_0, \mathsf{NIZK.Sim}_1)$ such that the advantage $\mathrm{Adv}^{\mathsf{zk}}_{\mathsf{NIZK},\mathsf{NIZK.Sim},\mathcal{B}}(\lambda)$ is negligible for all PPT adversaries $\mathcal{B}$. In Figure 6.2, we construct a simulator $\mathsf{Sim} = (\mathsf{Sim}_0, \mathsf{Sim}_1)$.

Let $\mathcal{A}$ be a PPT distinguisher between $\mathrm{Exp}^{(0)\text{-}\mathsf{enc}}_{\mathsf{dpiO},\mathcal{A}}(\lambda)$ and $\mathrm{Exp}^{(1)\text{-}\mathsf{enc}}_{\mathsf{Sim},\mathcal{A}}(\lambda)$ (making at most Q oracle queries for some polynomial Q). We prove indistinguishability between the real and the ideal distribution via a series of hybrids starting from the ideal distribution $\mathrm{Exp}^{(1)\text{-}\mathsf{enc}}_{\mathsf{Sim},\mathcal{A}}(\lambda)$.

GAME $\mathbf{G}_0$. This game is identical to $\mathrm{Exp}^{(1)\text{-enc}}_{\mathsf{Sim},\mathcal{A}}(\lambda)$. We remark that in this game, the tuple $(y; y')$ is produced using the adversarially chosen sampler S^{in} on input of the adversarially chosen state *state* and input x, where S^{in} is supplied with uniform randomness from $\{0,1\}^{k(\lambda)}$.

GAME $\mathbf{G}_1$. This game is identical to $\mathbf{G}_0$ except for the fact that for each query $(S^{\mathsf{in}}, \mathit{state}, x)$, the sampler S^{in} is supplied with randomness $G(r)$ for uniform r (instead of true randomness). Due to the strong regularity of G and by a standard hybrid argument over all queries, the statistical distance between $\mathbf{G}_0$ and $\mathbf{G}_1$ is negligible. More precisely, $|\Pr[\mathit{out}_1 = 1] - \Pr[\mathit{out}_0 = 1]| \leqslant Q \cdot \epsilon_{\mathsf{reg}}(\lambda)$, where Q is a polynomial upper bound for the number of oracle queries of $\mathcal{A}$.

$\mathsf{Sim}_0(1^\lambda)$

$(\sigma, \tau_\Pi) \leftarrow \mathsf{NIZK.Sim}_0(1^\lambda)$

$G \leftarrow \mathsf{ELF.Gen}(2^{k(\lambda)}, 2^{k(\lambda)})$

$pp := (\sigma, G)$

$td := \tau_\Pi$

return (pp, td)

(a) Definition of the first phase of the simulator Sim_0. Sim_0 generates a simulated CRS together with a corresponding trapdoor and sets up the parameters for the ELF in injective mode.

$\mathsf{Sim}_1(pp, td, S^{\mathsf{in}}, y, \mathit{state})$

parse $pp =: (\sigma, G)$

parse $td =: \tau_\Pi$

$\pi \leftarrow \mathsf{NIZK.Sim}_1(\sigma, \tau_\Pi, (G, S^{\mathsf{in}}, \mathit{state}, y))$

$\mathit{aux} := \pi$

return aux

(b) Definition of the second phase of the simulator Sim_1. Sim_1 uses the NIZK trapdoor τ_Π to simulate a proof π for y.

Figure 6.2: Definition of the simulator $\mathsf{Sim} = (\mathsf{Sim}_0, \mathsf{Sim}_1)$ in Figures 6.2a and 6.2b, respectively, for the proof of the simulatability of encodings property of dpiO.

GAME $\mathbf{G}_2$. This game is the same as $\mathbf{G}_1$ with the difference that σ is produced honestly using $\mathsf{NIZK.Setup}(1^\lambda)$. Additionally, for each adversarial query $(S^{\mathsf{in}}, \mathit{state}, x)$, the proof π is produced honestly by $\mathsf{NIZK.Prove}(\sigma, (G, S^{\mathsf{in}}, \mathit{state}, y), (y', x, r))$, where $G(r)$ are the random coins supplied to the sampler S^{in}. The view of $\mathcal{A}$ in game $\mathbf{G}_2$ is distributed exactly as in the real game $\mathrm{Exp}^{(0)\text{-enc}}_{\mathsf{dpiO},\mathcal{A}}(\lambda)$.

We construct a PPT adversary $\mathcal{B}_2$ for the zero-knowledge property of NIZK. Given a common reference string σ, $\mathcal{B}_2$ samples an ELF G in injective mode via $\mathsf{ELF.Gen}(2^{k(\lambda)}, 2^{k(\lambda)})$ and invokes $\mathcal{A}$ on input of $pp := (\sigma, G)$. Each time $\mathcal{A}$ queries his oracle on $(S^{\mathsf{in}}, \mathit{state}, x)$, $\mathcal{B}_2$ draws random coins $r \in \{0,1\}^{k(\lambda)}$ and invokes the sampler S^{in} on input of (state, x) with random coins $G(r)$ to obtain $(y; y')$. In order to produce π, $\mathcal{B}_2$ calls his prove oracle on input of $(G, S^{\mathsf{in}}, \mathit{state}, y)$ with witness (y', x, r). Therefore, if $\mathcal{B}_2$ is supplied with an honest CRS and honestly generated proofs, $\mathcal{B}_2$ perfectly simulates $\mathbf{G}_2$ for $\mathcal{A}$. Otherwise, if $\mathcal{B}_2$ is supplied with a simulated CRS and simulated proofs, $\mathcal{B}_2$ perfectly simulates $\mathbf{G}_1$. Hence, $|\Pr[\mathit{out}_2 = 1] - \Pr[\mathit{out}_1 = 1]| \leqslant \mathrm{Adv}^{\mathsf{zk}}_{\mathsf{NIZK},\mathsf{NIZK.Sim},\mathcal{B}_2}(\lambda)$.

Thus, we have

$$\begin{aligned}\mathrm{Adv}^{(0)\text{-enc}}_{\mathsf{dpiO},\mathsf{Sim},\mathcal{A}}(\lambda) &= \left|\Pr\left[\mathit{out}_2 = 1\right] - \Pr\left[\mathit{out}_0 = 1\right]\right| \\ &\leqslant Q \cdot \epsilon_{\mathsf{reg}}(\lambda) + \mathrm{Adv}^{\mathsf{zk}}_{\mathsf{NIZK},\mathsf{NIZK.Sim},\mathcal{B}}(\lambda)\end{aligned}$$

concluding the proof since both terms are negligible in λ.

RESTRICTED CORRECTNESS. Let $S^{in} \in \mathcal{S}^{in}$ be an input sampler, let $x_1, \ldots, x_\ell$ be arbitrary inputs from $L \cap \{0,1\}^\lambda$, and let C be a circuit from the family $\mathcal{C}_\lambda$. To prove that dpiO satisfies Definition 5.4, we prove that dpiO satisfies statistical closeness, computational correctness and is support respecting.

Statistical closeness. We begin with statistical closeness and proceed over a series of hybrids for that purpose.

GAME $\mathbf{G}_0$. This game is the ideal game $\mathrm{Exp}^{(1)\text{-close}}_{\mathsf{dpiO},S^{in},(x_1,\ldots,x_\ell),C,\mathcal{A}}(\lambda)$. Note that the sampler S^{in} is called using true randomness whereas the experiment $\mathrm{Exp}^{(0)\text{-close}}_{\mathsf{dpiO},S^{in},(x_1,\ldots,x_\ell),C,\mathcal{A}}(\lambda)$ generates samples using $G(r)$, where r is truly random.

GAME $\mathbf{G}_1$. This game is identical to $\mathbf{G}_0$ with the difference that each call of the sampler S^{in} is supplied with $G(r)$ as randomness (where r is sampled uniformly for each call). Due to the strong regularity of G, and by a hybrid argument over all ℓ calls of S^{in}, the statistical distance between $\mathbf{G}_1$ and $\mathbf{G}_0$ is negligible.

GAME $\mathbf{G}_2$. This game is the real game $\mathrm{Exp}^{(0)\text{-close}}_{\mathsf{dpiO},S^{in},(x_1,\ldots,x_\ell),C,\mathcal{A}}(\lambda)$. We now argue that the view of $\mathcal{A}$ in $\mathbf{G}_1$ is distributed identically to his view in $\mathbf{G}_2$. $\mathbf{G}_2$ samples public parameters pp via $\mathsf{Setup}(1^\lambda)$ and $\overline{S^{in}}$ via $\overline{S^{in}} \leftarrow \mathsf{Encode}(pp, S^{in})$. Further, (y_j, aux_j) are sampled as $state_j \leftarrow \mathsf{stf}(y_1, \ldots, y_{j-1}, \bot, \ldots, \bot)$ and $(y_j, aux_j, y'_j) \leftarrow \overline{S^{in}}(state_j, x_j)$, for $j \in [\ell]$. Let Γ be the obfuscation $\Gamma \leftarrow \mathsf{Obfuscate}(pp, S^{in}, C)$ of the circuit C with respect to sampler S^{in}. Due to the perfect correctness of iO, Γ computes the same functionality as $\overline{C}[\mathsf{stf}, (\sigma, G), S^{in}, C, K]$, where K is a freshly generated key for the PRF F. Hence, by the perfect completeness of NIZK, on input of $((y_1, aux_1), \ldots, (y_\ell, aux_\ell))$, Γ evaluates the circuit C on input of $(y_1, \ldots, y_\ell)$ with random coins $F(K, (y_1, \ldots, y_\ell))$. Therefore, the view of $\mathcal{A}$ in the games $\mathbf{G}_1$ and $\mathbf{G}_2$ only differs in the fact that $\mathbf{G}_1$ supplies C with true random coins whereas $\mathbf{G}_2$ supplies C with $F(K, (y_1, \ldots, y_\ell))$ as randomness. As F is a perfect PRF (see Definition 4.1), the distribution $\{K \leftarrow \mathsf{F.KGen}(1^\lambda) : F(K, (y_1, \ldots, y_\ell))\}$ is identical to the uniform distribution over the image of F. Therefore, the view of $\mathcal{A}$ in $\mathbf{G}_1$ and $\mathbf{G}_2$ is distributed identically.

Computational correctness. Computational correctness follows from the security of the PRF. Let C be a probabilistic circuit from $\mathcal{C}_\lambda$. Consider the following series of hybrids.

GAME $\mathbf{G}_0$. This game is identical to $\mathrm{Exp}^{(1)\text{-rcorr}}_{\mathsf{dpiO},C,\mathcal{A}}(\lambda)$.

GAME $\mathbf{G}_1$. Game $\mathbf{G}_1$ is identical to $\mathbf{G}_0$ except that initially, a PRF key K is sampled and the random coins for circuit C are computed as $F(K, (y_1, \ldots, y_\ell))$. Due to the security of F, we have

$$\left|\Pr\left[out_1 = 1\right] - \Pr\left[out_0 = 1\right]\right| \leqslant \mathrm{Adv}^{\mathsf{prf}}_{F,\mathcal{A}}(\lambda).$$

GAME $\mathbf{G}_2$. $\mathbf{G}_2$ is identical to $\mathrm{Exp}^{(0)\text{-rcorr}}_{\mathsf{dpiO},C,\mathcal{A}}(\lambda)$. Due to perfect correctness of iO and perfect completeness of NIZK, we have that $\Pr[out_1 = 1] = \Pr[out_2 = 1]$.

Support respecting. Since an encoded input sampler $\overline{\mathsf{S}^{\mathsf{in}}}$ produces y by evaluating S^{in} with random coins $G(r) \subseteq \{0,1\}^{k(\lambda)}$, we have that for all $pp \in \mathit{supp}(\mathsf{Setup}(1^\lambda))$, all $\overline{\mathsf{S}^{\mathsf{in}}} \leftarrow \mathsf{Encode}(pp, \mathsf{S}^{\mathsf{in}})$, all $x \in L$, all $\mathit{state} \in \mathcal{T}$, and all $(y, \mathit{aux}; \cdot) \in \mathit{supp}(\overline{\mathsf{S}^{\mathsf{in}}}(\mathit{state}, x))$, $(y; \cdot) \in \mathit{supp}(\mathsf{S}^{\mathsf{in}}(\mathit{state}, x))$.

Furthermore, by construction, perfect completeness of NIZK and perfect correctness of iO, for all $\Gamma \leftarrow \mathsf{Obfuscate}(pp, \mathsf{S}^{\mathsf{in}}, C)$, all $j \in [\ell]$, all inputs $(y_j, \mathit{aux}_j; \cdot) \in \mathit{supp}(\overline{\mathsf{S}^{\mathsf{in}}}(\mathit{state}_j, x_j))$, where $\mathit{state}_j \coloneqq \mathsf{stf}(y_1, \ldots, y_{j-1}, \bot, \ldots, \bot)$, we have that $\Gamma(y_1, \mathit{aux}_1, \ldots, y_\ell, \mathit{aux}_\ell) \in \mathit{supp}(C(y_1, \ldots, y_\ell))$.

HYBRID ARGUMENTS. In the following, we heavily use a standard technique called hybrid arguments. Computational indistinguishability of random variables can easily be seen to satisfy transitivity. That is, if the random variables X_1, X_2 and X_2, X_3 are pairwise computationally indistinguishable, then X_1 and X_3 are computationally indistinguishable. Basically, this follows from the triangle inequality. More precisely, for all PPT adversaries $\mathcal{A}$, $\Pr[\mathcal{A}(X_1) = 1] - \Pr[\mathcal{A}(X_2) = 1] \leqslant \epsilon_1(\lambda)$ and $\Pr[\mathcal{A}(X_2) = 1] - \Pr[\mathcal{A}(X_3) = 1] \leqslant \epsilon_2(\lambda)$ for some negligible functions $\epsilon_1(\lambda)$, $\epsilon_2(\lambda)$. Then,

$$\begin{aligned}
&\big|\Pr[\mathcal{A}(X_1) = 1] - \Pr[\mathcal{A}(X_3) = 1]\big| \\
&= \big|\Pr[\mathcal{A}(X_1) = 1] - \Pr[\mathcal{A}(X_2) = 1] + \Pr[\mathcal{A}(X_2) = 1] - \Pr[\mathcal{A}(X_3) = 1]\big| \\
&\leqslant \underbrace{\big|\Pr[\mathcal{A}(X_1) = 1] - \Pr[\mathcal{A}(X_2) = 1]\big|}_{\leqslant \epsilon_1(\lambda)} + \underbrace{\big|\Pr[\mathcal{A}(X_2) = 1] - \Pr[\mathcal{A}(X_3) = 1]\big|}_{\leqslant \epsilon_2(\lambda)}.
\end{aligned}$$

However, this transitivity does not immediately generalize to polynomially many "steps". This is because negligible functions can be somewhat contrived such that the sum of polynomially many negligible functions does not turn out to be negligible. For instance, the functions

$$\epsilon_i(\lambda) \coloneqq \begin{cases} 1 & \text{if } \lambda \leqslant i \\ 0 & \text{otherwise} \end{cases}$$

are negligible (for every $i \in \mathbb{N}$), whereas the function $f(\lambda) \coloneqq \sum_{i=1}^{\lambda} \epsilon_i(\lambda)$ is not negligible. In order to prove indistinguishability of polynomially many random variables (or security games), we use a standard proof technique called hybrid arguments. Basically, the above intuition to generalize the transitivity of indistinguishability to polynomially many steps works if each pairwise indistinguishability can be upper bounded by the *same* negligible function $\epsilon(\lambda)$. Then, intuitively, the triangle inequality yields

$$\begin{aligned}
&\big|\Pr\left[\mathcal{A}(X_1) = 1\right] - \Pr\left[\mathcal{A}(X_n) = 1\right]\big| \\
&\leqslant \sum_{i=1}^{n-1} \underbrace{\big|\Pr\left[\mathcal{A}(X_i = 1)\right] - \Pr\left[\mathcal{A}(X_{i+1} = 1)\right]\big|}_{\leqslant \epsilon(\lambda)} \leqslant (n-1)\epsilon(\lambda).
\end{aligned}$$

However, ensuring that all $\big|\Pr[\mathcal{A}(X_i = 1)] - \Pr[\mathcal{A}(X_{i+1} = 1)]\big|$ can be upper bounded by the same negligible function actually requires some extra work.

Suppose the indistinguishability between X_i and X_{i+1} can be based on the same decisional assumption such that a successful distinguisher between X_1 and X_n can be used to construct a single adversary $\mathcal{B}$ on that assumption. Suppose this decisional assumption states that the distributions Y_0 and Y_1 are indistinguishable. The distinguisher $\mathcal{B}$, on input of a challenge y from

Y_0 or Y_1, guesses an index $i \leftarrow [n-1]$, uses his y to sample x from the distribution X_i (if y is distributed according to Y_0) or from X_{i+1} (if y is distributed according to Y_1), and calls $\mathcal{A}$ on x. The advantage of $\mathcal{B}$ is

$$\begin{aligned}
&\Pr\left[\mathcal{B}(Y_0)=1\right]-\Pr\left[\mathcal{B}(Y_1)=1\right]\\
&=\sum_{j=1}^{n-1}\left(\Pr\left[\mathcal{B}(Y_0)=1\wedge i=j\right]-\Pr\left[\mathcal{B}(Y_1)=1\wedge i=j\right]\right)\\
&=\frac{1}{n-1}\cdot\sum_{j=1}^{n-1}\left(\Pr\left[\mathcal{B}(Y_0)=1\mid i=j\right]-\Pr\left[\mathcal{B}(Y_1)=1\mid i=j\right]\right)\\
&=\frac{1}{n-1}\cdot\sum_{j=1}^{n-1}\left(\Pr\left[\mathcal{A}(X_j)=1\right]-\Pr\left[\mathcal{A}(X_{j+1})=1\right]\right)\\
&=\frac{1}{n-1}\cdot\left(\Pr\left[\mathcal{A}(X_1)=1\right]-\Pr\left[\mathcal{A}(X_n)=1\right]\right).
\end{aligned}$$

This formally proves that X_1 and X_n are indistinguishable.

In order to avoid notational overhead, we will not make each reduction explicit.

SECURITY. Let $D \in \mathcal{C}^{\text{dyn-ind}}$ be an arbitrary dynamic-input indistinguishable circuit sampler over $\mathcal{C}$. Let p_C be an upper bound on the size of the circuits C, C' and C'' (including all hard-coded variables). We assume that all circuits to be obfuscated using iO are padded to size $p_C(\lambda)$.

To prove that dpiO satisfies indistinguishability (Definition 5.5), we proceed over a series of hybrids. Toward contradiction, we assume that there exists an adversary $\mathcal{A}$ of polynomial size s who distinguishes $\text{Exp}^{(0)\text{-ind}}_{\text{dpiO},D,\mathcal{A}}(\lambda)$ from $\text{Exp}^{(1)\text{-ind}}_{\text{dpiO},D,\mathcal{A}}(\lambda)$ with non-negligible advantage after making a polynomial number Q of queries to the oracle.

GAME $\mathbf{G}_0$. This game samples a random bit $b \leftarrow \{0,1\}$, and simulates the experiment $\text{Exp}^{(b)\text{-ind}}_{\text{dpiO},D,\mathcal{A}}(\lambda)$. More precisely, $\mathcal{A}$ has access to the public parameters pp and an oracle $\mathcal{O}^{(b)\text{-ind}}(D)$, which on input of an input sampler S^{in}, draws a sample (C_0, C_1, z) from D and outputs (C_0, C_1, z) together with an obfuscation $\text{Obfuscate}(pp, S^{\text{in}}, C_b)$. $\mathcal{A}$ outputs a guess b' and the game returns 1 if and only if $b' = b$. By assumption, the advantage of $\mathcal{A}$ in distinguishing $\text{Exp}^{(0)\text{-ind}}_{\text{dpiO},D,\mathcal{A}}(\lambda)$ from $\text{Exp}^{(1)\text{-ind}}_{\text{dpiO},D,\mathcal{A}}(\lambda)$ is non-negligible. Hence,

$$\begin{aligned}
\epsilon(\lambda) &:= \left|\Pr\left[out_0=1\right]-\frac{1}{2}\right|\\
&= \frac{1}{2}\cdot\left|\Pr\left[\text{Exp}^{(0)\text{-ind}}_{\text{dpiO},D,\mathcal{A}}(\lambda)=1\right]-\Pr\left[\text{Exp}^{(1)\text{-ind}}_{\text{dpiO},D,\mathcal{A}}(\lambda)=1\right]\right|
\end{aligned}$$

is a non-negligible function in λ.

GAME $\mathbf{G}_1$. This game is identical to $\mathbf{G}_0$ except for the fact that the ELF G is sampled in *extremely lossy mode*. More precisely, G is sampled via $\text{ELF.Gen}(2^{k(\lambda)}, t)$, where t is a polynomial such that every PPT algorithm of (circuit) size s has advantage at most $\epsilon/2$ in distinguishing $\text{ELF.Gen}(2^{k(\lambda)}, 2^{k(\lambda)})$ from $\text{ELF.Gen}(2^{k(\lambda)}, t)$. The advantage of $\mathcal{A}$ in $\mathbf{G}_1$ is therefore lower bounded by $\epsilon/2$, that is

$$\left|\Pr\left[out_1=1\right]-\frac{1}{2}\right| \geqslant \frac{\epsilon}{2}. \tag{6.1}$$

Otherwise, $\mathcal{A}$ has advantage greater than $\epsilon/2$ in distinguishing $\mathsf{ELF.Gen}(2^{k(\lambda)}, 2^{k(\lambda)})$ from $\mathsf{ELF.Gen}(2^{k(\lambda)}, t)$.

GAME $\mathbf{G}_2$. This game proceeds exactly as $\mathbf{G}_1$, except that after sampling $b \leftarrow \{0,1\}$, the game always sets up the experiment $\mathrm{Exp}^{(1)\text{-ind}}_{\mathsf{dpiO},D,\mathcal{A}}(\lambda)$, but still returns 1 if and only if $\mathcal{A}$'s guess b' equals b. Clearly,

$$\Pr\left[out_2 = 1\right] = \frac{1}{2} \tag{6.2}$$

and the advantage of $\mathcal{A}$ in $\mathbf{G}_2$ equals 0.

In the following, we prove that

$$\left|\Pr\left[out_2 = 1\right] - \Pr\left[out_1 = 1\right]\right| \leqslant Q \cdot negl(\lambda), \tag{6.3}$$

where *negl* is a negligible function and Q is a polynomial in λ. Equation (6.3) together with Equations (6.1) and (6.2) yields a contradiction to our assumption that ϵ is non-negligible as a function in λ.

In the following, S^{in}_q denotes the input sampler queried in the q-th oracle query. The circuits $C_{q,0}$ and $C_{q,1}$ denote the circuits sampled in the q-th oracle query using the circuit sampler D. Towards proving Equation (6.3), we define a sequence of hybrids as follows.

GAME $\mathbf{G}_{1.q}$. This game is identical to $\mathbf{G}_1$ except for the fact that the first $q-1$ oracle queries are answered using an obfuscation Γ_k of $C_{k,1}$ instead of $C_{k,b}$. Clearly, we have that

$$\Pr\left[out_1 = 1\right] = \Pr\left[out_{1.1} = 1\right] \tag{6.4}$$

$$\Pr\left[out_2 = 1\right] = \Pr\left[out_{1.Q+1} = 1\right] \tag{6.5}$$

Furthermore, we have that

$$\left|\Pr\left[out_{1.Q+1} = 1\right] - \Pr\left[out_{1.1} = 1\right]\right| \tag{6.6}$$

$$\leqslant \sum_{q=1}^{Q} \left|\Pr\left[out_{1.q} = 1\right] - \Pr\left[out_{1.q+1} = 1\right]\right|. \tag{6.7}$$

Due to Equations (6.4), (6.5) and (6.7), it suffices to upper bound the distinguishing gap between $\mathbf{G}_{1.q}$ and $\mathbf{G}_{1.q+1}$ in order to prove Equation (6.3).

Due to the statistical soundness of NIZK, with overwhelming probability over the random coins of Setup, for all $q \in [Q]$, there is no word $w \in \{0,1\}^* \setminus L^{G,S^{in}_q}_{state}$ such that a valid proof π for the statement $w \in L^{G,S^{in}_q}_{state}$ exists. Hence, every obfuscated circuit Γ_q, produced by the oracle on input of an input sampler S^{in}_q, simulates the randomized computation of the circuit $C_{q,b}$ only on well-formed inputs, i. e., on outputs of S^{in}_q using random coins from the range of G. Consequently, we only need to perform the hybrid argument over all circuit inputs from [CLTV15] restricted to well-formed inputs. Since ELF is in extremely lossy mode, this set of well-formed inputs is *extremely sparse*. Let

$$B_{q,j} \coloneqq \left\{ S^{in}_q\left(\mathsf{stf}(y_1, \ldots, y_{j-1}), x; G(r)\right) \;\middle|\; \begin{array}{l} x \in L \qquad , \; r \in \{0,1\}^{k(\lambda)} \\ k \in [j-1] \, , \, y_k \in B_{q,k} \end{array} \right\}$$

be this set of all well-formed inputs for input position $j \in [\ell]$. The set $B_{q,j}$ contains at most $|L| \cdot t^{j-1}$ elements. Further, let

$$\gamma_{q,1} < \cdots < \gamma_{q,\bar{t}}$$

be the ordered enumeration of all ℓ-tuples in $B_q := \prod_{j=1}^{\ell} B_{q,j}$, where $\bar{t}$ denotes the size of B_q.[29] Hence, the total number of well-formed inputs

$$\bar{t} := \prod_{j=1}^{\ell} |B_{q,j}| \leqslant \left(|L| \cdot t^{\ell-1}\right)^{\ell} \leqslant |L|^{\ell} \cdot t^{(\ell^2)}$$

is polynomial in λ (given that ℓ is a constant, and $|L|$ and t are polynomial).

[29] *We remark that the values of each set $B_{q,j}$ can be computed efficiently by evaluating S^{in} on all possible inputs and states from $L \times (\prod_{k=1}^{j-1} B_{q,k})$ and all possible random coins in the range of G. Note that since ELF is strongly regular, it is possible to enumerate the image of G in polynomial time.*

Towards proving indistinguishability between $\mathbf{G}_{1.q}$ and $\mathbf{G}_{1.q+1}$, we conduct a hybrid argument over all well-formed inputs for the obfuscation Γ_q and gradually replace the evaluation of circuit $C_{q,b}$ with $C_{q,1}$. From here on, our proof strategy is similar to the one employed in [CLTV15]. However, we only need to consider polynomially many hybrids, hence, this strategy only incurs a polynomial reduction loss relative to the underlying assumptions.

GAME $\mathbf{G}_{1.q.i}$. The game $\mathbf{G}_{1.q.i}$ is identical to game $\mathbf{G}_{1.q}$ except that the oracle answers the q-th query differently using an obfuscation of the circuit

$$\overline{C}'\left[\mathsf{stf}, (\sigma, G), S^{in}_q, C_{q,b}, C_{q,1}, K_q, \gamma_{q,i}\right],$$

defined in Figure 6.3, using iO.

Circuit $\overline{C}'[\mathsf{stf}, (\sigma, G), S^{in}, C_0, C_1, K, \gamma_i](x)$

parse $x =: ((y_1, aux_1), \ldots, (y_\ell, aux_\ell))$
$state_j := \mathsf{stf}(y_1, \ldots, y_{j-1}, \bot, \ldots, \bot)$
if $\neg\left(\forall j \in [\ell]\colon \mathsf{NIZK.Verify}(\sigma, (G, S^{in}, state_j, y_j), aux_j) = 1\right)$ **then**
  return $\bot$
$\gamma := (y_1, \ldots, y_\ell)$
if $\gamma < \gamma_i$ **then** $r := F(K, \gamma)$; **return** $C_1(\gamma; r)$
if $\gamma = \gamma_i$ **then** $r := F(K, \gamma)$; **return** $C_b(\gamma; r)$
if $\gamma > \gamma_i$ **then** $r := F(K, \gamma)$; **return** $C_b(\gamma; r)$

Figure 6.3: Definition of the circuit $\overline{C}'$.

The circuits

$$\overline{C}\left[\mathsf{stf}, (\sigma, G), S^{in}_q, C_{q,b}, K_q\right] \text{ and}$$
$$\overline{C}'\left[\mathsf{stf}, (\sigma, G), S^{in}_q, C_{q,b}, C_{q,1}, K_q, \gamma_{q,1}\right]$$

are functionally equivalent. On input of $x = ((y_1, aux_1), \ldots, (y_\ell, aux_\ell))$, both return $C_{q,b}(y_1, \ldots, y_\ell)$ with randomness $r = F(K_q, (y_1, \ldots, y_\ell))$. Hence, this game hop is justified by the security of iO. More formally, for every PPT adversary $\mathcal{A}$, there exists a functionally equivalent circuit sampler $D_{1.q.1}$ producing circuits as above and a PPT adversary $\mathcal{B}_{1.q.1}$ such that

$$\left|\Pr\left[out_{1.q} = 1\right] - \Pr\left[out_{1.q.1} = 1\right]\right| \leqslant \mathrm{Adv}^{\text{io-ind}}_{iO, D_{1.q.1}, \mathcal{B}_{1.q.1}}(p_C(\lambda)). \quad (6.8)$$

Furthermore, let $\gamma_{q,\bar{t}+1} > \gamma_{q,\bar{t}}$, then the circuits

$$\overline{C}'\left[\mathsf{stf}, (\sigma, G), S_q^{\mathsf{in}}, C_{q,b}, C_{q,1}, K_q, \gamma_{q,\bar{t}+1}\right] \text{ and}$$
$$\overline{C}\left[\mathsf{stf}, (\sigma, G), S_q^{\mathsf{in}}, C_{q,1}, K_q\right]$$

are functionally equivalent. Hence, for every PPT adversary $\mathcal{A}$, there exists a functionally equivalent circuit sampler $D_{1.q+1}$ producing circuits as above and a PPT adversary $\mathcal{B}_{1.q+1}$, such that

$$\left|\Pr\left[out_{1.q.\bar{t}+1} = 1\right] - \Pr\left[out_{1.q+1} = 1\right]\right| \leqslant \mathsf{Adv}^{\mathsf{io\text{-}ind}}_{\mathsf{iO},D_{1.q+1},\mathcal{B}_{1.q+1}}(p_C(\lambda)). \tag{6.9}$$

Furthermore, we have that

$$\left|\Pr\left[out_{1.q.\bar{t}+1} = 1\right] - \Pr\left[out_{1.q.1} = 1\right]\right| \leqslant \sum_{i=1}^{\bar{t}} \left|\Pr\left[out_{1.q.i+1} = 1\right] - \Pr\left[out_{1.q.i} = 1\right]\right| \tag{6.10}$$

Due to Equations (6.8), (6.9) and (6.10), it suffices to upper bound the distinguishing gap between $\mathbf{G}_{1.q.i}$ and $\mathbf{G}_{1.q.i+1}$ in order to prove indistinguishability between $\mathbf{G}_{1.q}$ and $\mathbf{G}_{1.q+1}$.

For this purpose, we aim to exploit the dynamic-input indistinguishability of the circuit sampler D. Therefore, we first need to supply the evaluation of $C_{q,b}$ (and $C_{q,1}$) at input of γ_i with true randomness. Hence, we define another series of hybrids between $\mathbf{G}_{1.q.i}$ and $\mathbf{G}_{1.q.i+1}$.

GAME $\mathbf{G}_{1.q.i.1}$. This game is identical to $\mathbf{G}_{1.q.i}$ except for the fact that we use a punctured PRF key $K_q\{\gamma_{q,i}\}$ produced via $\mathsf{F.Punct}(K_q, \gamma_{q,i})$ and obfuscate the circuit

$$\overline{C}''\left[\mathsf{stf}, (\sigma, G), C_{q,b}, C_{q,1}, K_q\{\gamma_{q,i}\}, Y \coloneqq C_{q,b}(\gamma_{q,i}; \mathsf{F}(K_q, \gamma_{q,i})), \gamma_{q,i}\right]$$

as defined in Figure 6.4 using iO.

As F preserves the functionality under punctured keys, the circuits

$$\overline{C}'\left[\mathsf{stf}, (\sigma, G), S_q^{\mathsf{in}}, C_{q,b}, C_{q,1}, K_q, \gamma_{q,i}\right] \text{ and}$$
$$\overline{C}''\left[\mathsf{stf}, (\sigma, G), S_q^{\mathsf{in}}, C_{q,b}, C_{q,1}, K_q\{\gamma_{q,i}\}, \underbrace{Y}_{\coloneqq C_{q,b}(\gamma_{q,i}; \mathsf{F}(K_q, \gamma_{q,i}))}, \gamma_{q,i}\right]$$

are functionally equivalent. On input of $\gamma_{q,i}$, the circuits used in $\mathbf{G}_{1.q.i}$ outputs $C_{q,b}(\gamma_{q,i})$ using randomness $r \coloneqq \mathsf{F}(K_q, \gamma_{q,i})$, and the circuit used in $\mathbf{G}_{1.q.i.1}$ outputs Y which is computed as $Y \coloneqq C_{q,b}(\gamma_{q,i}; \mathsf{F}(K_q, \gamma_{q,i}))$. On all remaining inputs, these circuits also behave identically as F preserves functionality under punctured keys. Hence, for all PPT adversaries $\mathcal{A}$, there exists a functionally equivalent circuit sampler $D_{1.q.i.1}$ producing circuits as above distributed as in $\mathbf{G}_{1.q.i}$ and $\mathbf{G}_{1.q.i.1}$, respectively, and a PPT adversary $\mathcal{B}_{1.q.i.1}$ such that

$$\left|\Pr\left[out_{1.q.i} = 1\right] - \Pr\left[out_{1.q.i.1} = 1\right]\right| \leqslant \mathsf{Adv}^{\mathsf{io\text{-}ind}}_{\mathsf{iO},D_{1.q.i.1},\mathcal{B}_{1.q.i.1}}(p_C(\lambda)). \tag{6.11}$$

Circuit $\overline{C}''[\mathsf{stf}, (\sigma, G), S^{\mathsf{in}}, C_0, C_1, K\{\gamma_i\}, Y, \gamma_i](x)$

parse $x =: ((y_1, aux_1), \ldots, (y_\ell, aux_\ell))$

$state_j \coloneqq \mathsf{stf}(y_1, \ldots, y_{j-1}, \bot, \ldots, \bot)$

if $\neg\left(\forall j \in [\ell]\colon \mathsf{NIZK.Verify}(\sigma, (G, S^{\mathsf{in}}, state_j, y_j), aux_j) = 1\right)$ **then**

  return $\bot$

$\gamma \coloneqq (y_1, \ldots, y_\ell)$

if $\gamma < \gamma_i$ **then** $r \coloneqq F(K\{\gamma_i\}, \gamma)$; **return** $C_1(\gamma; r)$

if $\gamma = \gamma_i$ **then** **return** Y

if $\gamma > \gamma_i$ **then** $r \coloneqq F(K\{\gamma_i\}, \gamma)$; **return** $C_b(\gamma; r)$

Figure 6.4: Definition of the circuit $\overline{C}''$.

We note that the view of $\mathcal{A}$ in game $\mathbf{G}_{1.q.i.1}$ does not depend on the PRF key K_q (only on the punctured PRF key $K_q\{\gamma_{q,i}\}$). This enables a reduction to the selective security of F.

GAME $\mathbf{G}_{1.q.i.2}$. In this game we replace the randomness $F(K_q, (\gamma_{q,i}))$ by true randomness, i. e., we produce Y as $Y \coloneqq C_{q,b}(\gamma_{q,i}; R)$ for uniformly random $R \leftarrow \{0,1\}^{k(\lambda)}$. This game hop is justified by the selective PRF property. More formally, for every PPT adversary $\mathcal{A}$, there exists a PPT adversary $\mathcal{B}_{1.q.i.2}$, such that

$$\left|\Pr\left[out_{1.q.i.2} = 1\right] - \Pr\left[out_{1.q.i.1} = 1\right]\right| \leqslant \mathrm{Adv}^{\text{s-pprf}}_{F,\mathcal{B}_{1.q.i.2}}(\lambda). \tag{6.12}$$

GAME $\mathbf{G}_{1.q.i.3}$. This game is the same as $\mathbf{G}_{1.q.i.2}$ except for the fact that Y is produced using the circuit $C_{q,1}$, i. e., $Y \coloneqq C_{q,1}(\gamma_{q,i}; R)$. This game hop is justified by the fact that the circuit sampler D is a dynamic-input indistinguishable sampler. More precisely, for every PPT adversary $\mathcal{A}$ there exists a PPT adversary $\mathcal{B}_{1.q.i.3}$ such that

$$\left|\Pr\left[out_{1.q.i.3} = 1\right] - \Pr\left[out_{1.q.i.2} = 1\right]\right| \leqslant \mathrm{Adv}^{\text{dyn-ind}}_{D,\mathcal{B}_{1.q.i.3}}(\lambda). \tag{6.13}$$

GAME $\mathbf{G}_{1.q.i.4}$. This game is the same as $\mathbf{G}_{1.q.i.3}$ with the difference that we again use pseudorandom coins to compute Y, i. e., $Y \coloneqq C_{q,1}(\gamma_{q,i}; F(K_q, \gamma_{q,i}))$. For every PPT adversary $\mathcal{A}$, there exists a PPT adversary $\mathcal{B}_{1.q.i.4}$ such that

$$\left|\Pr\left[out_{1.q.i.4} = 1\right] - \Pr\left[out_{1.q.i.3} = 1\right]\right| \leqslant \mathrm{Adv}^{\text{s-pprf}}_{F,\mathcal{B}_{1.q.i.4}}(\lambda). \tag{6.14}$$

Since F preserves functionality under punctured keys, the two circuits

$$\overline{C}''\left[\mathsf{stf}, (\sigma, G), S^{\mathsf{in}}_q, C_{q,b}, C_{q,1}, K_q\{\gamma_{q,i}\}, \underbrace{Y}_{:=C_{q,1}(\gamma_{q,i}; F(K_q, \gamma_{q,i}))}, \gamma_{q,i}\right]$$

$$\overline{C}'\left[\mathsf{stf}, (\sigma, G), S^{\mathsf{in}}_q, C_{q,b}, C_{q,1}, K_q, \gamma_{q,i+1}\right]$$

are functionally equivalent. Therefore, for all PPT avdersaries $\mathcal{A}$, there exists a functionally equivalent circuit sampler $D_{1.q.i+1}$ producing circuits as above

distributed as in $\mathbf{G}_{1.q.i.4}$ and $\mathbf{G}_{1.q.i+1}$, respectively, and a PPT adversary $\mathcal{B}_{1.q.i+1}$ such that

$$\left|\Pr\left[out_{1.q.i+1}=1\right]-\Pr\left[out_{1.q.i.4}=1\right]\right| \leqslant \mathrm{Adv}^{\text{io-ind}}_{\mathsf{iO},\mathsf{D}_{1.q.i+1},\mathcal{B}_{1.q.i+1}}(p_C(\lambda)). \tag{6.15}$$

Combining Equations (6.11), (6.12), (6.13), (6.14) and (6.15), we obtain

$$\left|\Pr\left[out_{1.q.i+1}=1\right]-\Pr\left[out_{1.q.i}=1\right]\right| \leqslant 2\cdot\mathrm{Adv}^{\text{io-ind}}_{\mathsf{iO},\mathsf{D}',\mathcal{B}'}(p_C(\lambda))+2\cdot\mathrm{Adv}^{\text{s-pprf}}_{\mathsf{F},\mathcal{B}''}(\lambda)+\mathrm{Adv}^{\text{dyn-ind}}_{\mathsf{D},\mathcal{B}'''}(\lambda) \tag{6.16}$$

for PPT adversaries $\mathcal{B}'$, $\mathcal{B}''$, $\mathcal{B}'''$ and a functionally equivalent circuit sampler D'. Since iO is a secure indistinguishability obfuscator, F is a puncturable PRF and D is a dynamic-input indistinguishable sampler, the above quantities are negligible in λ.

Combining Equations (6.8), (6.9), (6.10) and (6.16), the advantage in distinguishing $\mathbf{G}_{1.q}$ and $\mathbf{G}_{1.q+1}$ can be upper bounded by

$$\left|\Pr\left[out_{1.q+1}=1\right]-\Pr\left[out_{1.q}=1\right]\right| \leqslant |L|^{\ell}\cdot t^{\ell^2}\cdot negl(\lambda)+2\cdot\mathrm{Adv}^{\text{io-ind}}_{\mathsf{iO},\mathsf{D}',\mathcal{B}'}(p_C(\lambda)) \tag{6.17}$$

for some PPT adversary $\mathcal{B}'$ and some functionally equivalent circuit sampler D', which is negligible.

Hence, combining Equations (6.4), (6.5), (6.7) and (6.17), the advantage to distinguish $\mathbf{G}_1$ and $\mathbf{G}_2$ is upper bounded by

$$\left|\Pr\left[out_2=1\right]-\Pr\left[out_1=1\right]\right| \leqslant Q\cdot negl'(\lambda)+2\cdot\mathrm{Adv}^{\text{io-ind}}_{\mathsf{iO},\mathsf{D}',\mathcal{B}'}(p_C(\lambda)),$$

for some negligible function $negl'$, for some PPT adversary $\mathcal{B}'$ and some functionally equivalent circuit sampler D', which is negligible. This proves Equation (6.3).

Finally, together with Equations (6.1) and (6.2), we reach a contradiction concluding the proof.

□

EXTENSION. We note that our notion of doubly-probabilistic IO can be extended to support a set of independent public parameters such that encoded input samplers may depend on different public parameters. Suppose $(\mathsf{Setup}, \mathsf{Encode}, \mathsf{Obfuscate})$ is a 5-source dpIO scheme. Then, the obfuscation $\mathsf{Obfuscate}(pp_1[1\text{-}3], pp_2[4,5], S^{\mathsf{in}}, C)$ of the circuit C requires the first three inputs to be sampled with an encoded sampler with respect to pp_1 and the last two inputs to be sampled with an encoded sampler with respect to pp_2.

The above prove can be directly extended to this setting by choosing each extremely-lossy mode such that $\mathcal{A}$ has advantage at most $\epsilon/2\ell$ in distinguishing the lossy from the injective mode. For the applications covered in this thesis, we do not need this extension and refer the reader to the original publication [ACH20] for more details.

7 LEVELED HOMOMORPHIC ENCRYPTION

In this chapter, we demonstrate that our notion of dpIO from Chapter 5 can be applied to construct leveled homomorphic encryption in a similar way as in [CLTV15]. This construction leads to a transformation which operates on an encryption scheme E, satisfying IND-CPA security (and possibly other security properties, e. g., KDM security), and produces a leveled homomorphic encryption scheme that retains the security properties of E. We emphasize that since our notion of dpIO achieves security against dynamic-input indistinguishable samplers, we do not need to assume any additional properties for E. This is a further improvement compared to [CLTV15], where additional structural properties such as lossiness or re-randomizability are required.[30]

Let $\mathcal{P} = (\mathcal{P}_{\lambda,\widetilde{L}})_{\lambda,\widetilde{L}\in\mathbb{N}}$ be a family of sets of polynomial sized depth-$\widetilde{L}$ circuits of arity k, i. e., the set $\mathcal{P}_{\lambda,\widetilde{L}}$ contains circuits of polynomial size in λ of depth $\widetilde{L}$. Further, for any $\lambda, \widetilde{L} \in \mathbb{N}$ the circuits in $\mathcal{P}_{\lambda,\widetilde{L}}$ share the common input domain $\{0,1\}^k$.

[30] *It is not known whether such structural properties can be achieved from IO in conjunction with unstructured assumptions, such as one-way functions.*

DEFINITION 7.1 (Leveled homomorphic encryption [Gen09]). A *leveled homomorphic encryption (LHE) scheme* for message space $\{0,1\}$ is a family of encryption schemes $\mathsf{LHE} := \{\mathsf{E}_{\widetilde{L}} = (\mathsf{KGen}_{\widetilde{L}}, \mathsf{Enc}_{\widetilde{L}}, \mathsf{Dec}_{\widetilde{L}}, \mathsf{Eval}_{\widetilde{L}})\}_{\widetilde{L}\in\mathbb{N}}$ such that each $\mathsf{E}_{\widetilde{L}}$ is an IND-CPA secure public-key encryption scheme for message space $\{0,1\}$ which is homomorphic for all polynomial-size depth-$\widetilde{L}$ circuits. That is, each $\mathsf{E}_{\widetilde{L}}$ allows to homomorphically and compactly evaluate any polynomial-size depth-$\widetilde{L}$ circuit via an evaluation algorithm $\mathsf{Eval}_{\widetilde{L}}$ of size polynomial in $(\lambda, \widetilde{L})$, and all the $\mathsf{E}_{\widetilde{L}}$ use the same decryption circuit $\mathsf{Dec}_{\widetilde{L}} = \mathsf{Dec}$. An LHE scheme is required to satisfy the following notion of correctness.

CORRECTNESS. For all $\lambda \in \mathbb{N}$, all $\widetilde{L} \in \mathbb{N}$, all $m_1, \ldots, m_k \in \mathcal{M}_\lambda$, $P \in \mathcal{P}_{\lambda,\widetilde{L}}$,

$$\Pr\left[\begin{array}{rcll} (pk, ek, sk) & \leftarrow & \mathsf{KGen}_{\widetilde{L}}(1^\lambda) & \mathsf{Dec}(sk, c) \\ c_i & \leftarrow & \mathsf{Enc}_{\widetilde{L}}(pk, m_i) & : \quad = \\ c & \leftarrow & \mathsf{Eval}_{\widetilde{L}}(ek, P, c_1, \ldots, c_k) & P(m_1, \ldots, m_k) \end{array}\right] = 1.$$

IND-CPA SECURITY. For all $\widetilde{L} \in \mathbb{N}$, all PPT adversaries $\mathcal{A}$,

$$\mathrm{Adv}^{\text{ind-cpa}}_{\mathsf{LHE},\mathcal{A}}(\lambda) := \left| \Pr\left[\begin{array}{rcl} (pk, ek, sk) & \leftarrow & \mathsf{KGen}_{\widetilde{L}}(1^\lambda) \\ c^* & \leftarrow & \mathsf{Enc}_{\widetilde{L}}(pk, 0) \end{array} : \mathcal{A}(pk, ek, c^*) = 1\right] - \Pr\left[\begin{array}{rcl} (pk, ek, sk) & \leftarrow & \mathsf{KGen}_{\widetilde{L}}(1^\lambda) \\ c^* & \leftarrow & \mathsf{Enc}_{\widetilde{L}}(pk, 1) \end{array} : \mathcal{A}(pk, ek, c^*) = 1\right] \right|$$

is negligible in λ.

COMPACTNESS. The size of the output of $\mathsf{Eval}|_{\widetilde{L}}$ is polynomial in $(\lambda, \widetilde{L})$ and independent of the size of the circuit P.

Note that Definition 7.1 distinguishes between an evaluation key *ek* and a public (encryption) key *pk*. The evaluation key can equivalently be defined to be part of the public key *pk* as in Definition 2.24.

REMARK 7.1. Note that a leveled (or fully) homomorphic encryption scheme for message space $\{0,1\}$ implies a leveled (or fully) homomorphic encryption scheme for message space $\{0,1\}^\lambda$.

Let stf be the trivial state function, i. e., $\mathsf{stf}\colon (y_1, y_2) \mapsto \bot$ for each tuple $(y_1, y_2) \in (\{0,1\}^{n(\lambda)} \cup \{\bot\})^2$. Let $\mathsf{E} = (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ be an arbitrary IND-CPA secure public-key encryption scheme. Let the input sampler class $\mathcal{S}^{\mathsf{in}}$

$$\mathcal{S}^{\mathsf{in}} := \left\{ S^{\mathsf{in}}_{pk} \;\middle|\; (pk, \cdot) \in \mathit{supp}(\mathsf{E.KGen}(1^\lambda)) \right\},$$

where S^{in}_{pk} is defined in Figure 7.1. For technical reasons, the input length of S^{in}_{pk} is required to match the security parameter λ (see Definition 5.1). Hence, the input domain L of the input samplers in $\mathcal{S}^{\mathsf{in}}$ equals $\{0^\lambda, 1^\lambda\}$. That is, $\mathcal{S}^{\mathsf{in}}$

$S^{\mathsf{in}}_{pk}(\mathit{state}, x)$

$y := \bot, y' := \bot$

$$y \leftarrow \begin{cases} \mathsf{E.Enc}(pk, 0) & \text{, if } x = 0^\lambda \\ \mathsf{E.Enc}(pk, 1) & \text{, if } x = 1^\lambda \end{cases}$$

return $(y; y')$

Figure 7.1: Input samplers used for the construction of LHE from dpIO.

contains all input samplers S^{in}_{pk} which on input of a state *state* and an input $x \in L$, sample a ciphertext y encrypting x via $y := \mathsf{E.Enc}(pk, x)$ and $y' := \bot$ ignoring *state*, where *pk* is a public key in the range of $\mathsf{E.KGen}(1^\lambda)$. However, for a more convenient notation, we will slightly abuse notation and treat S^{in}_{pk} as if its input domain L equals $\{0,1\}$.

Further, let $\mathcal{C}$ be the class of polynomial-size randomized circuits and let $\mathcal{C}^{\mathsf{dyn\text{-}ind}}$ be the class of dynamic-input indistinguishable samplers over $\mathcal{C}$.

THEOREM 7.1. *Let* $(\mathsf{Setup}, \mathsf{Encode}, \mathsf{Obfuscate})$ *be a 2-source* dpiO *scheme for* $(\mathsf{stf}, \mathcal{S}^{\mathsf{in}}, \mathcal{C}, \mathcal{C}^{\mathsf{dyn\text{-}ind}})$ *and let* E *be an IND-CPA secure public-key encryption scheme. Then,* LHE *as defined in Figure 7.2 is an IND-CPA secure LHE scheme.*

The proof strategy is similar as in [CLTV15]. On a high level, we aim to reduce the IND-CPA security of LHE to the IND-CPA security of the underlying encryption scheme E. However, the evaluation key *ek* contains information (even though obfuscated) about the secret decryption keys of each level. For the purpose of invoking the security of E on the challenge ciphertext, we need to remove this dependency on sk_0. Therefore, we gradually (starting from level $\widetilde{L}$) replace the obfuscations of the circuits C with obfuscations of *trapdoor* circuits tC which simply output samples produced by the encoded sampler $\overline{S^{\mathsf{in}}_{pk}}(0)$ on the fixed input 0 (hence, not needing any information on decryption keys). Let $p_{\mathsf{lhe}}(\lambda)$ denote an upper bound on the size of C and tC including all hard-coded values during the proof. These two circuits sample

LHE.KGen$(1^\lambda, 1^{\widetilde{L}})$

for $i \in \{0, \dots, \widetilde{L}\}$ **do**
  $(pk_i, sk_i) \leftarrow \mathsf{E.KGen}(1^\lambda)$
  $pp_i \leftarrow \mathsf{Setup}(1^{p_{\mathsf{lhe}}(\lambda)})$
  $\overline{S^{\mathsf{in}}_{pk_i}} \leftarrow \mathsf{Encode}(pp_i, S^{\mathsf{in}}_{pk_i})$
for $i \in \{1, \dots, \widetilde{L}\}$ **do**
  $\Gamma_i \leftarrow \mathsf{Obfuscate}\left(pp_{i-1}, S^{\mathsf{in}}_{pk_{i-1}}, C[\overline{S^{\mathsf{in}}_{pk_i}}, sk_{i-1}]\right)$
$pk := \overline{S^{\mathsf{in}}_{pk_0}}$, $sk := sk_{\widetilde{L}}$
$ek := (\Gamma_1, \dots, \Gamma_{\widetilde{L}})$
return (pk, ek, sk)

(a) The key generation algorithm producing a public encryption key pk, a secret decryption key sk and an evaluation key ek. The circuit C is defined in Figure 7.3.

LHE.Enc$(pk, m \in \{0,1\})$

parse $pk =: \overline{S^{\mathsf{in}}_{pk_0}}$
$(y, aux, y') \leftarrow \overline{S^{\mathsf{in}}_{pk_0}}(\bot, m)$
return $c := (y, aux)$

(b) The encryption algorithm sampling via the encoded input sampler for level 0.

LHE.Dec(sk, c)

parse $sk =: sk_{\widetilde{L}}$
parse $c =: (y, aux)$
return $\mathsf{E.Dec}(sk_{\widetilde{L}}, y)$

(c) The decryption algorithm using the key for level $\widetilde{L}$.

LHE.Eval$(ek, P, (c_1, \dots, c_k))$

for $i \in \{1, \dots, \widetilde{L}\}$ **do**
  foreach gate g on level i **do**
    // let α_g, β_g denote the inputs for gate g
    $\gamma_g := \Gamma_i(\alpha_g, \beta_g)$
  // let g_{out} denote the output gate of P
return $\gamma_{g_{\mathsf{out}}}$

(d) The evaluation algorithm using the obfuscated circuits Γ_i to homomorphically evaluate circuit P on the ciphertexts $c_1, \dots, c_k$.

Figure 7.2: Instantiation of an LHE scheme LHE from some PKE scheme E using a 2-source dpIO scheme dpiO.

from the same encoded sampler $\overline{S^{\mathsf{in}}_{pk}}$ but behave differently in which inputs they supply $\overline{S^{\mathsf{in}}_{pk}}$ with. Due to the simulatability of encodings property of dpiO and the IND-CPA security of E, these two circuits are dynamic-input indistinguishable. Hence, by the indistinguishability property of dpiO for $\mathcal{C}^{\text{dyn-ind}}$, an honest evaluation key and an evaluation key consisting only of trapdoor circuits are indistinguishable.

Given these modifications, the challenge ciphertext c^* consists of an encryption of a bit b under pk_0 accompanied by some auxiliary information produced by the corresponding encoded sampler. This auxiliary information might leak information on the bit b and thereby prevents to directly employ the IND-CPA security of E. However, as dpiO satisfies simulatability of encodings, this auxiliary information can be simulated without knowledge of b and, hence, contains no information about b. Therefore, by the IND-CPA security of E, LHE is IND-CPA secure.

$\mathsf{C}[\overline{\mathsf{S}^{\mathsf{in}}_{pk}}, sk'](x_\alpha, x_\beta)$

$\alpha \leftarrow \mathsf{E.Dec}(sk', x_\alpha)$

$\beta \leftarrow \mathsf{E.Dec}(sk', x_\beta)$

$(y, aux, y') \leftarrow \overline{\mathsf{S}^{\mathsf{in}}_{pk}}(\bot, \alpha \mathbin{\overline{\wedge}} \beta)$

return (y, aux)

(a) Definition of the probabilistic circuit C which implements homomorphic evaluation of a single $\overline{\wedge}$-gate.

$\mathsf{tC}[\overline{\mathsf{S}^{\mathsf{in}}_{pk}}](x_\alpha, x_\beta)$

$(y, aux, y') \leftarrow \overline{\mathsf{S}^{\mathsf{in}}_{pk}}(\bot, 0)$

return (y, aux)

(b) Definition of the trapdoor analogue tC of the circuit C which is unknowing of any secret decryption keys.

Figure 7.3: Definition of the circuits C and tC.

Proof of Theorem 7.1. We prove that LHE satisfies correctness and IND-CPA security. Let P be a polynomial sized circuit of depth $\widetilde{L}$ which consists only of $\overline{\wedge}$-gates[31] with input space $\{0,1\}^k$.

[31] *This is without loss of generality since $\overline{\wedge}$-gates are a functionally complete set of Boolean operators.*

CORRECTNESS. We prove that on every level i, every gate g in P is evaluated exactly as the original gate given the decrypted inputs.

Let $i \in [\widetilde{L}]$ be a level and let g be a gate on level i. Further, let $(x_\alpha, aux_\alpha, \cdot)$ be in the support of $\overline{\mathsf{S}^{\mathsf{in}}_{pk_{i-1}}}(\bot, \alpha)$ and $(x_\beta, aux_\beta, \cdot)$ be in the support of $\overline{\mathsf{S}^{\mathsf{in}}_{pk_{i-1}}}(\bot, \beta)$. By perfect correctness of E, the circuit $\mathsf{C}[\overline{\mathsf{S}^{\mathsf{in}}_{pk_i}}, sk_{i-1}]$ on input of (x_α, aux_α) and (x_β, aux_β) always outputs an element from the support of $\overline{\mathsf{S}^{\mathsf{in}}_{pk_i}}(\bot, \alpha \mathbin{\overline{\wedge}} \beta)$.[32] Furthermore, since dpiO is support respecting (Definition 5.4), both the encoded sampler $\overline{\mathsf{S}^{\mathsf{in}}}$ and the obfuscated circuit Γ respect the support of the original sampler S^{in} and the original circuit $\mathsf{C}[\overline{\mathsf{S}^{\mathsf{in}}_{pk_i}}, sk_{i-1}]$, respectively. Hence, the obfuscation Γ of $\mathsf{C}[\overline{\mathsf{S}^{\mathsf{in}}_{pk_i}}, sk_{i-1}]$ (with respect to $\mathsf{S}^{\mathsf{in}}_{pk_{i-1}}$) on input of (x_α, aux_α) and (x_β, aux_β) always outputs an element from the support of $\mathsf{S}^{\mathsf{in}}_{pk_i}(\bot, \alpha \mathbin{\overline{\wedge}} \beta)$. Hence, the probability that gate g is not evaluated correctly is zero.

[32] *We implicitly use that the supports of $\overline{\mathsf{S}^{\mathsf{in}}}(\bot, 0)$ and $\overline{\mathsf{S}^{\mathsf{in}}}(\bot, 1)$ are disjoint.*

SECURITY. To prove that LHE is IND-CPA secure, we proceed over a series of hybrids. Let $\mathcal{A}$ be a PPT adversary. Further, let $p_{\mathsf{lhe}}(\lambda)$ denote an upper bound on the size of C and tC including all hard-coded values during the proof.

GAME $\mathbf{G}^b_0$. This game is exactly the IND-CPA game for LHE, where the challenge ciphertext c^* contains the bit b.

GAME $\mathbf{G}^b_1$. This game is the same as $\mathbf{G}^b_0$ with the difference that the evaluation key consists of obfuscations $t\Gamma_i$ of the circuit $\mathsf{tC}[\overline{\mathsf{S}^{\mathsf{in}}_{pk_i}}]$.

CLAIM 7.1. For every PPT adversary $\mathcal{A}$, we have that

$$\left|\Pr\left[out^b_1 = 1\right] - \Pr\left[out^b_0 = 1\right]\right| \leqslant negl(\lambda)$$

for some negligible function $negl$.

Proof. We define a series of $\widetilde{L}+1$ hybrid games $\mathbf{G}^b_{0.i}$ for $i \in \{0, \ldots, \widetilde{L}\}$ as follows.

GAME $\mathbf{G}^b_{0.i}$. $\mathbf{G}^b_{0.i}$ is defined in Figure 7.4. Hence, $\mathbf{G}^b_{0.i}$ is identical to $\mathbf{G}^b_{0.i-1}$

$\mathbf{G}^{b}_{0.i}$

for $i \in \{0, \ldots, \widetilde{L}\}$ **do**
  $(pk_i, sk_i) \leftarrow \mathsf{E.KGen}(1^\lambda)$
  $pp_i \leftarrow \mathsf{Setup}(1^\lambda)$
  $\overline{S^{in}_{pk_i}} \leftarrow \mathsf{Encode}(pp_i, S^{in}_{pk_i})$
for $j \in \{1, \ldots, \widetilde{L} - i\}$ **do**
  $\Gamma_j \leftarrow \mathsf{Obfuscate}\left(pp_{j-1}, S^{in}_{pk_{j-1}}, C[\overline{S^{in}_{pk_j}}, sk_{j-1}]\right)$
for $j \in \{\widetilde{L} - i + 1, \ldots, \widetilde{L}\}$ **do**
  $t\Gamma_j \leftarrow \mathsf{Obfuscate}\left(pp_{j-1}, S^{in}_{pk_{j-1}}, tC[\overline{S^{in}_{pk_j}}]\right)$
$pk := \overline{S^{in}_{pk_0}}, sk := sk_{\widetilde{L}}$
$ek := \left(\Gamma_1, \ldots, \Gamma_{\widetilde{L}-i}, t\Gamma_{\widetilde{L}-i+1}, \ldots, t\Gamma_{\widetilde{L}}\right)$
return (pk, ek, sk)

Figure 7.4: Definition of $\mathbf{G}^{b}_{0.i}$.

with the difference that the $\widetilde{L} - i + 1$-th obfuscated circuit in the evaluation key ek is produced as

$$\Gamma_{\widetilde{L}-i+1} \leftarrow \mathsf{Obfuscate}\left(pp_{\widetilde{L}-i}, S^{in}_{pk_{\widetilde{L}-i}}, C\left[\overline{S^{in}_{pk_{\widetilde{L}-i+1}}}, sk_{\widetilde{L}-i}\right]\right)$$

in $\mathbf{G}^{b}_{0.i-1}$, and as

$$t\Gamma_{\widetilde{L}-i+1} \leftarrow \mathsf{Obfuscate}\left(pp_{\widetilde{L}-i}, S^{in}_{pk_{\widetilde{L}-i}}, tC\left[\overline{S^{in}_{pk_{\widetilde{L}-i+1}}}\right]\right)$$

in $\mathbf{G}^{b}_{0.i}$. By definition, we have

$$\Pr\left[out^{b}_{0.0} = 1\right] = \Pr\left[out^{b}_{0} = 1\right] \quad \text{and} \tag{7.1}$$

$$\Pr\left[out^{b}_{0.\widetilde{L}} = 1\right] = \Pr\left[out^{b}_{1} = 1\right]. \tag{7.2}$$

CLAIM 7.2. For every PPT adversary $\mathcal{A}$, there exists a dynamic-input indistinguishable sampler $D^{b}_{0.i+1}$ and a PPT adversary $\mathcal{B}^{b}_{0.i+1}$, such that

$$\left|\Pr\left[out^{b}_{0.i} = 1\right] - \Pr\left[out^{b}_{0.i+1} = 1\right]\right| \leqslant \mathrm{Adv}^{\mathsf{ind}}_{\mathsf{dpiO}, D^{b}_{0.i+1}, \mathcal{B}^{b}_{0.i+1}}(p_{\mathsf{lhe}}(\lambda)).$$

By a standard hybrid argument and Equations (7.1) and (7.2), we get $|\Pr[out^{b}_{0} = 1] - \Pr[out^{b}_{1} = 1]| \leqslant \widetilde{L} \cdot \mathrm{Adv}^{\mathsf{ind}}_{\mathsf{dpiO}, D, \mathcal{B}}(p_{\mathsf{lhe}}(\lambda))$ for some dynamic-input indistinguishable sampler D and some PPT adversary $\mathcal{B}$. By the security of dpiO, this quantity is negligible.

Proof of Claim 7.2. We first show that there exists a PPT adversary $\mathcal{B}^{b}_{0.i+1}$ and the circuit sampler $D^{b}_{0.i+1}$ defined in Figure 7.5, such that the distinguishing gap between $\mathbf{G}^{b}_{0.i}$ and $\mathbf{G}^{b}_{0.i+1}$ is upper bounded by the advantage of $\mathcal{B}^{b}_{0.i+1}$ in the dpiO security game. Then we show that $D^{b}_{0.i+1}$ is a dynamic-input indistinguishable sampler.

We construct a PPT adversary $\mathcal{B}^{b}_{0.i+1}$ for the security game of dpiO as follows. $\mathcal{B}^{b}_{0.i+1}$ receives as input public parameters pp', the circuits C_0 and C_1

$\mathsf{D}^{b}_{0.i+1}(1^\lambda)$

$pp \leftarrow \mathsf{Setup}(1^\lambda)$

$(pk, sk) \leftarrow \mathsf{E.KGen}(1^\lambda)$

$(pk', sk') \leftarrow \mathsf{E.KGen}(1^\lambda)$

$\overline{S^{in}_{pk}} \leftarrow \mathsf{Encode}(pp, S^{in}_{pk})$

$C_0 \coloneqq C\left[\overline{S^{in}_{pk}}, sk'\right]$

$C_1 \coloneqq tC\left[\overline{S^{in}_{pk}}\right]$

$z \coloneqq (pk, pk', sk', pp)$

return (C_0, C_1, z)

Figure 7.5: Circuit sampler used in the proof of IND-CPA security of LHE.

together with auxiliary information z produced by $\mathsf{D}^{b}_{0.i+1}$, and an obfuscation $\overline{\Gamma} \leftarrow \mathsf{Obfuscate}(pp', S^{in}_{pk'}, C_{\widetilde{b}})$ for some $\widetilde{b} \in \{0,1\}$. Initially, $\mathcal{B}^{b}_{0.i+1}$ samples parameters as in $\mathbf{G}^{b}_{0.i}$ embedding his input as $pp_{\widetilde{L}-i-1} \coloneqq pp'$, $pk_{\widetilde{L}-i-1} \coloneqq pk'$, and $pk_{\widetilde{L}-i} \coloneqq pk$, and defines the public key to be $pk \coloneqq \overline{S^{in}_{pk_0}}$ for $\overline{S^{in}_{pk_0}} \leftarrow \mathsf{Encode}(pp_0, S^{in}_{pk})$. $\mathcal{B}^{b}_{0.i+1}$ produces the obfuscations Γ_j for $j \in \{1, \ldots, \widetilde{L}-i-1\}$ and $t\Gamma_j$ for $j \in \{\widetilde{L}-i+1, \ldots, \widetilde{L}\}$ as in $\mathbf{G}^{b}_{0.i}$ and defines the evaluation key as $ek \coloneqq (\Gamma_1, \ldots, \Gamma_{\widetilde{L}-i-1}, \overline{\Gamma}, t\Gamma_{\widetilde{L}-i+1}, \ldots, t\Gamma_{\widetilde{L}})$.

$\mathcal{B}^{b}_{0.i+1}$ produces the challenge ciphertext c^* for the bit b as $c^* \leftarrow \overline{S^{in}_{pk_0}}(b)$. Finally, $\mathcal{B}^{b}_{0.i+1}$ calls $\mathcal{A}$ on input (pk, ek, c^*) and outputs $\mathcal{A}$'s output. If $\widetilde{b} = 0$, $\mathcal{B}^{b}_{0.i+1}$ perfectly simulates $\mathbf{G}^{b}_{0.i}$ for $\mathcal{A}$, else $\mathcal{B}^{b}_{0.i+1}$ perfectly simulates $\mathbf{G}^{b}_{0.i+1}$ for $\mathcal{A}$. Hence,

$$\left|\Pr\left[out^{b}_{0.i+1} = 1\right] - \Pr\left[out^{b}_{0.i} = 1\right]\right| \leqslant \mathrm{Adv}^{\mathsf{ind}}_{\mathsf{dpiO}, \mathsf{D}^{b}_{0.i+1}, \mathcal{B}^{b}_{0.i+1}}(p_{\mathsf{lhe}}(\lambda)).$$

What remains to prove is that $\mathsf{D}^{b}_{0.i+1}$ is a dynamic-input indistinguishable sampler.

CLAIM 7.3. The circuit sampler $\mathsf{D}^{b}_{0.i+1}$ as defined in Figure 7.5 is a dynamic-input indistinguishable sampler.

Proof of Claim 7.3. We prove this using a sequence of two hybrids. Let $\mathcal{A}$ be a PPT adversary in the dynamic-input indistinguishability game for the circuit sampler $\mathsf{D}^{b}_{0.i+1}$.

GAME $\mathbf{H}_0$. This is the dynamic-input indistinguishability game as defined in Figure 2.2a. In detail, $\mathbf{H}_0$ samples (C_0, C_1, z) according to $\mathsf{D}^{b}_{0.i+1}$, gives C_0, C_1 together with the state z to $\mathcal{A}$ who adaptively chooses an input x. $\mathbf{H}_0$ evaluates the circuit $C_{b'}$ (for a random bit b') at input of x (using fresh random coins) and sends the output to $\mathcal{A}$.

GAME $\mathbf{H}_1$. By the simulatability of encodings property of dpiO, there exists a simulator $\mathsf{Sim} = (\mathsf{Sim}_0, \mathsf{Sim}_1)$, such that outputs of the encoded sampler (including the auxiliary information *aux*) and outputs of the original sampler together with simulated auxiliary information are indistinguishable. The game $\mathbf{H}_1$ is the same as $\mathbf{H}_0$ with the difference that the public parameters are simulated as $(pp, td) \leftarrow \mathsf{Sim}_0(1^\lambda)$. Given the adversarially chosen input

$x = (x_\alpha, x_\beta)$, $\mathbf{H}_1$ computes $(m_0, m_1) := (\alpha \overline{\wedge} \beta, 0)$ using the secret key sk' (which is part of z). Furthermore, $\mathbf{H}_1$ produces $(y, \cdot) \leftarrow S^{\mathsf{in}}_{pk}(\bot, m_{b'})$ and $aux \leftarrow \mathsf{Sim}_1(td, S^{\mathsf{in}}_{pk}, y, \bot)$.

In order to upper bound the distinguishing gap between $\mathbf{H}_1$ and $\mathbf{H}_0$, we construct an adversary $\mathcal{B}_{\mathbf{H}.1}$ for the simulatability of encodings property of dpiO. On input of public parameters pp, $\mathcal{B}_{\mathbf{H}.1}$ uses these as public parameters and simulates $\mathbf{H}_1$ for $\mathcal{A}$. However, instead of sampling y and aux with $S^{\mathsf{in}}_{pk}(\bot, m_{b'})$ and $\mathsf{Sim}_1(td, S^{\mathsf{in}}_{pk}, y, \bot)$, respectively, $\mathcal{B}_{\mathbf{H}.1}$ calls the encoding oracle provided by the simulatability game on input of $(S^{\mathsf{in}}_{pk}, \bot, m_{b'})$ to obtain y and aux. If $\mathcal{B}_{\mathbf{H}.1}$ interacts with $\mathrm{Exp}^{(0)\text{-enc}}_{\mathsf{dpiO},\mathcal{B}_{\mathbf{H}.1}}(p_{\mathsf{lhe}}(\lambda))$, $\mathcal{B}_{\mathbf{H}.1}$ perfectly simulates $\mathbf{H}_0$ for $\mathcal{A}$. Otherwise, if $\mathcal{B}_{\mathbf{H}.1}$ interacts with $\mathrm{Exp}^{(1)\text{-enc}}_{\mathsf{dpiO},\mathcal{B}_{\mathbf{H}.1}}(p_{\mathsf{lhe}}(\lambda))$, $\mathcal{B}_{\mathbf{H}.1}$ perfectly simulates $\mathbf{H}_1$ for $\mathcal{A}$. Hence, we have

$$\left|\Pr\left[out_{\mathbf{H}_1} = 1\right] - \Pr\left[out_{\mathbf{H}_0} = 1\right]\right| \leqslant \mathrm{Adv}^{\mathsf{enc}}_{\mathsf{dpiO},\mathsf{Sim},\mathcal{B}_{\mathbf{H}.1}}(p_{\mathsf{lhe}}(\lambda))$$

which is negligible by the simulatability of encodings property of dpiO.

The advantage of $\mathcal{A}$ in game $\mathbf{H}_1$ is negligible. To realize that, we construct an adversary $\mathcal{D}_{\mathbf{H}.1}$ for the IND-CPA security of E with respect to the public key pk. On input of pk, $\mathcal{D}_{\mathbf{H}.1}$ samples (C_0, C_1, z) in the same way as in $\mathbf{H}_1$ (embedding pk) and calls $\mathcal{A}$ on input of (C_0, C_1, z) to obtain (x, st). $\mathcal{D}_{\mathbf{H}.1}$ decrypts $x = (x_\alpha, x_\beta)$ to obtain α and β using sk', and outputs $(m_0, m_1) := (\alpha \overline{\wedge} \beta, 0)$ to the IND-CPA game. In return, $\mathcal{D}_{\mathbf{H}.1}$ receives a ciphertext c^*, simulates the corresponding auxiliary information $aux \leftarrow \mathsf{Sim}_1(pp, td, S^{\mathsf{in}}_{pk}, c^*, \bot)$, and invokes $\mathcal{A}$ on input of his previous state st and (c^*, aux). If c^* encrypts $m_{\widetilde{b}}$, y is distributed exactly as in $\mathbf{H}_1$ conditioned on $b' = \widetilde{b}$. Finally, $\mathcal{D}_{\mathbf{H}.1}$ outputs $\mathcal{A}$'s output. Hence, by the IND-CPA security of E,

$$\left|\Pr\left[out_{\mathbf{H}_1} = 1\right] - \frac{1}{2}\right| \leqslant \frac{1}{2} \cdot \mathrm{Adv}^{\mathsf{ind\text{-}cpa}}_{\mathsf{E},\mathcal{D}_{\mathbf{H}.1}}(\lambda)$$

which is negligible. This concludes the proof of Claim 7.3.

□

Together with Claim 7.3, this concludes the proof of Claim 7.2. □

Hence, we have that $|\Pr[out^b_1 = 1] - \Pr[out^b_0 = 1]|$ is negligible in λ, what was to be demonstrated. □

Towards being able to conduct a reduction to the IND-CPA security of E for the challenge ciphertext, we need to ensure that the auxiliary information which accompanies the ciphertext from $S^{\mathsf{in}}_{pk_0}(b)$ does not leak any information about the plaintext b. This is formalized via another game transition.

GAME $\mathbf{G}^b_2$. This game is identical to $\mathbf{G}^b_1$ except for the fact that the public parameters pp_0 are simulated by Sim_0 (additionally yielding a trapdoor td_0). Furthermore, the challenge ciphertext c^* is produced by drawing a sample $(y; y')$ from $S^{\mathsf{in}}_{pk_0}(b)$ and simulating the auxiliary information aux using $\mathsf{Sim}_1(pp_0, td_0, S^{\mathsf{in}}_{pk_0}, y, \bot)$. We recall that for every PPT adversary $\mathcal{A}$ such a simulator $\mathsf{Sim} = (\mathsf{Sim}_0, \mathsf{Sim}_1)$ exists as dpiO satisfies simulatability of encodings. Hence, there exists a PPT adversary $\mathcal{B}^b_2$, such that the distinguishing gap

$$\left|\Pr\left[out^b_2 = 1\right] - \Pr\left[out^b_1 = 1\right]\right| \leqslant \mathrm{Adv}^{\mathsf{enc}}_{\mathsf{dpiO},\mathsf{Sim},\mathcal{B}^b_2}(p_{\mathsf{lhe}}(\lambda))$$

which, by simulatability of encodings, is negligible in λ.

It remains to argue, that the games $\mathbf{G}_2^0$ and $\mathbf{G}_2^1$ are indistinguishable. This can be reduced to the IND-CPA security of E with respect to pk_0 as in $\mathbf{G}_2^b$ neither the corresponding secret key sk_0 nor the plaintext b are necessary for simulation.

CLAIM 7.4. For every PPT adversary $\mathcal{A}$, the distinguishing gap $|\Pr[out_2^0 = 1] - \Pr[out_2^1 = 1]|$ is negligible.

Proof. We construct a PPT adversary $\mathcal{B}_2'$ against the IND-CPA security of E. On input of pk, $\mathcal{B}_2'$ defines $pk_0 := pk$, samples key pairs $(pk_j, sk_j) \leftarrow \mathsf{E.KGen}(1^\lambda)$ for $j \in \{1, \ldots, \widetilde{L}\}$ and produces ek as in $\mathbf{G}_2$. $\mathcal{B}_2'$ outputs the messages 0 and 1 to the IND-CPA game and in turn receives a ciphertext c'^*. Furthermore, $\mathcal{B}_2'$ simulates the auxiliary information by $aux^* \leftarrow \mathsf{Sim}_1(pp_0, td_0, S^{\mathsf{in}}_{pk_0}, c'^*, \bot)$. Finally, $\mathcal{B}_2'$ invokes $\mathcal{A}$ on input of $(pk_0, ek, (c'^*, aux^*))$ and outputs $\mathcal{A}$'s output. If c'^* encrypts 0, $\mathcal{B}_2'$ perfectly simulates $\mathbf{G}_2^0$ for $\mathcal{A}$, and if c'^* encrypts 1, $\mathcal{B}_2'$ perfectly simulates $\mathbf{G}_2^1$ for $\mathcal{A}$. Hence,

$$\left|\Pr\left[out_2^0 = 1\right] - \Pr\left[out_2^1 = 1\right]\right| \leqslant \mathrm{Adv}^{\mathsf{ind\text{-}cpa}}_{\mathsf{E},\mathcal{B}_2'}(\lambda),$$

which is negligible by the IND-CPA security of E. □

□

Together with Theorem 6.1 from Chapter 6, we obtain the following corollary.

COROLLARY 7.1. *Assuming polynomially secure indistinguishability obfuscation and extremely lossy functions, then there exists a leveled homomorphic encryption scheme.*

Note that IND-CPA secure schemes, as required in our construction, can be constructed from (polynomially secure) IO and one-way functions (the latter being implied by ELFs). Previously, constructions of LHE were only known from the learning with error assumption, or from *subexponentially secure* indistinguishability obfuscation (together with lossy encryption, which can be based, for instance, on DDH, see also Section 9.5).

FULLY HOMOMORPHIC ENCRYPTION. [CLTV15] provide a generic LHE-to-FHE transformation. We briefly sketch the strategy. If an LHE scheme can handle a (slightly) super-polynomial number of levels $\widetilde{L}$, then it is fully homomorphic for all polynomial-size circuits. However, this would require generating and storing a super-polynomial number of evaluation keys $(ek_1, \ldots, ek_{\widetilde{L}})$. To overcome this issue, [CLTV15] (inspired by [BGL+15]) use an obfuscated master key generation program MProg, which generates the evaluation keys *on the fly*: on input of i, MProg returns ek_i. It was observed in [CLTV15] that pIO cannot be directly used to obfuscate this program, because the coins used by MProg must be correlated between two consecutive levels (for example, in our LHE construction, ek_i is an obfuscated program with a secret key sk_i and a public key pk_{i+1} hard-coded; hence, ek_{i+1} must contain the matching secret key for pk_{i+1}). Nevertheless, "de-randomizing" this program, using the standard technique of generating the coins using a pseudorandom function F (the coins for sk_i are generated as $\mathsf{F}(K, i)$, and the

coins for pk_{i+1} are generated as $F(K, i+1)$, which guarantees the appropriate correlation between the coins used in ek_i and ek_{i+1}), and obfuscating with standard IO, allows to prove security of this approach.

Applying this generic transformation, we obtain the following corollary.

COROLLARY 7.2. *Assuming slightly super-polynomially secure indistinguishability obfuscation and extremely lossy functions, there exists a fully homomorphic encryption scheme.*

Previously, constructions of FHE were only known from circular security assumptions over lattice-based cryptosystems, or subexponentially secure indistinguishability obfuscation and lossy encryption.

Applying the LHE-toFHE transformation from [CLTV15] requires to use super-polynomially secure IO. However, using the same techniques as we used to build dpIO, we can actually base this transformation on polynomially secure IO and ELFs.

In the original publication [ACH20], we provide an LHE-to-FHE transformation using the techniques developed in this part which only requires polynomially secure IO and ELFs. This is not part of this thesis.

Part II

INSTANTIATING THE ALGEBRAIC GROUP MODEL FROM OBFUSCATION

8 INTRODUCTION

In this part, we present the results of [AHK20]. This part is taken verbatim from [AHK20] with minor changes.

THE GENERIC GROUP MODEL. In order to analyze the plausibility and relative strength of computational assumptions in cyclic groups, Shoup [Sho97] and Maurer [Mau05] have proposed the *generic group model* (GGM). In the GGM, any adversary can only interact with the modeled group through an oracle. In particular, all computations in that group must be explicitly expressed in terms of the group operation. To prevent an adversary from locally performing computations, that adversary gets to see only truly random strings (in [Sho97]) or independent handles (in [Mau05]) as representations of group elements.[33]

The discrete logarithm and even many Diffie-Hellman-style problems are hard generically (i. e., when restricting group operations in the above way) [Sho97; MW98]. Hence, the only way to break such a generically hard assumption in a concrete group is to use the underlying group representation in a nontrivial way. In that sense, the GGM can be very useful as a sanity check for the validity of a given assumption, or even the security of a given cryptographic scheme. However, generic groups cannot be implemented: there exist cryptographic schemes that are secure in the GGM, but insecure when instantiated with *any* concrete group [Den02].

[33] *Other black-box abstractions of groups with similar ramifications exist [Nec94; BL96].*

THE ALGEBRAIC GROUP MODEL. The algebraic group model (AGM), introduced in [FKL18], is a relaxation of the GGM that tries to avoid impossibilities as in [Den02] while preserving the GGM's usefulness. Specifically, the AGM only considers *algebraic* (rather than generic) adversaries. An algebraic adversary $\mathcal{A}$ can make arbitrary use of the representation of group elements, but must supply an explicit decomposition for any of its output group elements in terms of input group elements. In other words, $\mathcal{A}$ must also output an explanation of how any group element in its output was computed from its input using the group operation.

Now [FKL18] show that many GGM proofs only use this type of algebraicity of an adversary, and carry over to the AGM. At the same time, GGM impossibilities like [Den02] do not apply to the AGM, since algebraic adversaries are able to work with the actual group (and not only with random or abstract representations of group elements).

THE AGM AND KNOWLEDGE-TYPE ASSUMPTIONS. The AGM is closely related to the notions of knowledge-type assumptions and extractability. To illustrate, assume that for any (possibly non-algebraic) adversary $\mathcal{A}$, we can find an extractor $\mathcal{E}$ that manages to extract from $\mathcal{A}$ a decomposition of $\mathcal{A}$'s output in terms of $\mathcal{A}$'s input. Then, composing $\mathcal{E}$ and $\mathcal{A}$ yields an algebraic adversary $\mathcal{A}_{\mathsf{alg}}$. In this situation, we can then say that without loss of generality, any adversary can be assumed to be algebraic.[34] Conversely,

[34] *This observation about algebraic adversaries has already been made in [BV98; PV05]. Also, similar but more specific knowledge-type assumptions have been used to prove concrete cryptographic constructions secure, e. g., [Dam92; HT98; BP04; Den06].*

any algebraic adversary by definition yields the results of such an extraction in its output.

This observation also provides a blueprint to *instantiating* the AGM: simply prove that any adversary $\mathcal{A}$ can be replaced by an algebraic adversary $\mathcal{A}_{\mathsf{alg}}$, possibly using an extraction process as above. If this extraction requires $\mathcal{A}$'s code and randomness but no other trapdoor, we obtain an AGM instantiation based on a knowledge-type assumption such as the knowledge-of-exponent assumption [Dam92]. Indeed, this was recently done by [KP19] under a very strong generalized version of the knowledge-of-exponent assumption [WS07]. Unfortunately, such knowledge-type assumptions are not falsifiable in the sense of Naor [Nao03] making them (arguably) undesirable for use in cryptography. Furthermore, it is not entirely clear how to assess the plausibility of such a universal and strong knowledge-type assumption. Naturally, the question arises whether an AGM implementation inherently requires such strong and non-falsifiable assumptions. Or, more generally:

Can we achieve knowledge-type properties from falsifiable assumptions?

Note that in the AGM, the discrete logarithm assumption implies the existence of extractable one-way functions (EOWFs) with unbounded auxiliary input. The existence of such EOWFs, however, conflicts with the existence of indistinguishability obfuscation [BCPR16]. Due to this barrier, we can only hope for an instantiation of some suitably relaxed variant of the AGM from falsifiable assumptions.

OUR STRATEGY: PRIVATE EXTRACTION. Since extraction as defined in the AGM inherently requires knowledge-type assumptions, we need to conceive a different extraction mechanism. Namely, we show that it is possible to extract a decomposition of $\mathcal{A}$'s outputs *from these outputs* if a suitable (secret) extraction trapdoor is known. In other words, our idea is to avoid non-falsifiable knowledge-type assumptions by assuming that extraction requires a special trapdoor which can be generated alongside the public parameters of the group. This entails a number of technical difficulties (see below), but allows us to rely entirely on falsifiable assumptions.

Specifically, our main result is an *algebraic wrapper* that transforms a given cyclic group into a new one which allows for an extraction of representations. More specifically, an element of the new group carries an encrypted representation of this group element relative to a fixed set of group elements (henceforth denoted "basis"). The group operation guarantees that newly produced group elements carry decompositions which are consistent with the decompositions contained in the input group elements. Further, the corresponding decryption key (which can be generated alongside the public parameters) serves as a trapdoor which allows to extract such decompositions.

OUR RESULTS. Our algebraic wrapper allows us to retrieve several AGM results (from [FKL18; FPS20]) in the standard model, in the sense that the group can be concretely implemented from falsifiable assumptions.[35] In particular, we show that in the algebraic wrapper,

[35] *Note that by "standard model", we mean that the group itself is formulated without idealizations and can be concretely implemented. While our construction itself does not rely on the ROM, we still can transfer some ROM proofs in the AGM to ROM proofs using our concrete group instantiation. We stress that a standard model instantiation of the (full-fledged) AGM from very strong* non-falsifiable *assumptions is already known due to [KP19].*

(i) the discrete logarithm assumption, the computational Diffie-Hellman assumption, the square Diffie-Hellman assumption, and the linear-combination Diffie-Hellman assumption (see [FKL18]) are all equivalent,

(ii) the security of the Schnorr signature scheme [Sch91] can be *tightly* reduced to the discrete logarithm assumption escaping impossibility results due to [FJS19],

(iii) the IND-CCA2 security of Schnorr-signed ElGamal [Jak98; TY98] can be *tightly* reduced to the discrete logarithm assumption.[36]

We stress that Items i and ii are not the contribution of this author and are hence not included in this thesis. Item iii is detailed in Chapter 13 and the conception and instantiation of the algebraic wrapper are detailed in Chapters 11 and 12, respectively. We refer the reader to the original publication [AHK20] for more details on Items i and ii.

While, on a technical level, the AGM proofs from [FKL18; FPS20] need to be adapted by adding some preparatory steps, the general AGM proof strategies can be replicated.

[36] *Tight security reductions provide a tight relation between the security of cryptographic schemes and the hardness of computational problems. Apart from their theoretical importance, tight reductions are also beneficial for practice, since they allow smaller keylength recommendations.*

LIMITATIONS. We note that not all known AGM proofs can be transported to the standard model using the above strategy. For instance, [FKL18] also prove the Boneh-Lynn-Shacham (BLS) [BLS04] signature scheme tightly secure in the AGM. Their reduction relies on the fact that the view of a signature forger is statistically independent of how simulated signatures are prepared by the reduction. However, with our algebraic wrapper, group elements (and thus BLS signatures) always carry an encrypted representation. This representation allows to infer information on how these group elements were generated. This additional information is problematic in the AGM-based proof of BLS signatures from [FKL18]. We believe it is an interesting open problem to obtain a tight security proof for the BLS scheme with our group.[37]

Furthermore, the amount of information which can be carried by a group element is limited by the size of that group element. In particular, in settings in which no a-priori bound on the size of a desired algebraic representation is known, our techniques do not apply. This can be problematic, e. g., for constructions that depend on q-type assumptions.

[37] *Note that impossibility results for tight reductions of schemes like BLS (e. g., [Coroo]) do not apply in our case, as the representation of our group elements is not unique.*

OUR ASSUMPTIONS. Our algebraic wrapper relies on a strong but falsifiable computational assumption: the existence of subexponentially strong indistinguishability obfuscation.[38] Additionally, we assume a re-randomizable encryption scheme. Together with subexponential IO, this implies a number of other strong primitives that we use: a variant of statistically correct probabilistic IO (see [CLTV15] and Chapter 10), fully homomorphic encryption (see [CLTV15]), and dual-mode non-interactive zero-knowledge (see [HU19]).

[38] *We note that IO and certain knowledge-type assumptions (which require universal extraction) contradict each other [BCPR16]. However, we stress that the notion of* private *extractability we obtain does* not *contradict IO.*

INTERPRETATION. Due to their inefficiency, we view algebraic wrappers not as a tool to obtain practical cryptographic primitives. Rather, we believe that algebraic wrappers show that the AGM is a useful abstraction and proof strategies developed for the AGM can in fact be replicated in the standard model, and even without resorting to knowledge-type assumptions.

ON IMPLEMENTING IDEALIZED MODELS. Replacing idealized (heuristic) models with concrete standard-model implementations is a widely studied intriguing problem. A well-known example for this is the line of work on programmable hash functions. A programmable hash function due to [HK12] is a cryptographic primitive which can be used to replace random oracles in several cryptographic schemes. Following their introduction, a line of work [FHPS13; HSW13; HSW14] leveraged multilinear maps or indistinguishability obfuscation to transport proofs from the random oracle model to the standard model. Our results can be interpreted as following this endeavor by leveraging indistinguishability obfuscation to replace the AGM with a standard model implementation (from falsifiable assumptions). From this angle, our algebraic wrapper relates to the AGM as programmable hash functions relate to the ROM.

8.1 TECHNICAL OVERVIEW

ALGEBRAIC WRAPPERS. We employ the notion of *group schemes* [AFH+16] (also called encoding schemes in [GGH13]) as a generalization of groups with potentially non-unique encodings of group elements. In such a group scheme, a dedicated algorithm is required to determine if two given group element encodings are equal. That is, the group is defined as the quotient set of all well-formed encodings modulo the equivalence relation induced by the equality test. Our algebraic wrapping process takes group parameters of $\mathbb{G}$ (which we refer to as "base group") as input, and outputs group parameters of a new group scheme $\mathbb{H}$ which allows for an efficient extraction process. To be specific, every $\mathbb{H}$-element $\widehat{h}$ can be viewed as a $\mathbb{G}$-element $h \in \mathbb{G}$, plus auxiliary information *aux*.

The core advantage of group schemes with non-unique encodings lies in the ability to let group elements carry such auxiliary information *aux*. In our algebraic wrapper, *aux* contains encrypted information expressing h as a linear combination of fixed base group elements $b_1, \ldots, b_n \in \mathbb{G}$. The corresponding decryption key (which can be generated alongside the group parameters) allows to extract this information, and essentially yields similar information any algebraic adversary (in the sense of the AGM) would have to provide for any output group element. However, we are facing a number of technical problems:

(a) Upon creation of each new group element encodings, a valid representation with respect to the basis elements $b_1, \ldots, b_n$ needs to be embedded.

(b) It should be possible to unnoticeably switch the basis elements $b_1, \ldots, b_n$ to an application-dependent basis. (For instance, in reductions to the discrete logarithm problem $dlog_g(h)$, one would desire to set the basis to g, h.)

(c) To preserve tightness of reductions from the AGM (which is necessary for some of our applications), it should be possible to re-randomize group element encodings *statistically*.

Our solution largely follows the group scheme from [AFH+16]. In particular, we address Item a by requiring each group element encoding $\widehat{h} \in \mathbb{H}$ to carry

an encrypted linear combination $z_1, \ldots, z_n$ of $\mathbb{G} \ni h = \sum_i b_i^{z_i}$ relative to the basis and force validity of this linear combination using a consistency proof. Consequently, sampling of new group elements requires explicit knowledge of such a linear combination. This is also true for the group operation. More precisely, the group operation may produce the ciphertext for the new group element encoding via homomorphic evaluation of the input ciphertexts but is required to extract the contained linear combination to produce the consistency proof. Note that the adversary must be able to evaluate the group operation and needs to produce corresponding consistency proofs. This process, however, particularly the involved random coins must not be under adversarial control. Thus, the group operation is obfuscated using probabilistic IO.

For Item b, we adapt a "switching" lemma from [AFH+16]. In [AFH+16], that lemma allows to switch between two different representations of the same group element, but under a fixed basis. In our case, we show that similar techniques allow to additionally switch the group elements that form this basis. This issue is dealt with by additional linear randomization of the base group elements. Furthermore, since the basis is computationally hidden, sampling group element encodings which are tailored to contain a certain linear combination requires knowledge of a secret trapdoor.

Note that the switching property already implies a notion of computational re-randomizability. For Item c, we introduce a re-randomization lemma using techniques from Item b in conjunction with a novel notion for *statistically correct* probabilistic IO.

At this point, one main conceptual difference to the line of work [AFH+16; AH18; FHHL18; AMP20] is that the basis elements $b_1, \ldots, b_n$ appear as part of the functionality of the new group scheme $\mathbb{H}$, not only in a proof.

Another main conceptual difference to [AFH+16; AH18; FHHL18; AMP20] is the notion of statistical re-randomizability of group elements. The group schemes from [AFH+16; AH18; FHHL18; AMP20] do not satisfy this property. This will be resolved by developing a stronger notion of *statistically correct* probabilistic IO which may be of independent interest.

APPLICATIONS. The applications of our algebraic wrapper have already been considered for the AGM in [FKL18; FPS20]. Hence, in this description, we focus on the technical differences that our extraction approach entails for these proofs. Further, we focus on the applications from [AHK20] which are the contributions of this author.

Recall that in the AGM by [FKL18], an adversary outputs an algebraic representation of each output group element to the basis of its input group elements. Therefore, the choice of this basis depends on the application.

The authors of [FPS20] prove the tight security of Schnorr-signed ElGamal based on the discrete logarithm assumption (in the AGM and ROM). Schnorr-signed ElGamal uses a public key of the form $(g, Y \coloneqq g^y)$, where y is the corresponding secret key. Encryption chooses a random exponent x and uses a (random oracle) hash $H'(Y^x)$ to blind the message. Further, a Schnorr signature relative to the public key (g, g^x) is used to certify g^x. This procedure ensures plaintext-awareness [BR95]. Hence, a ciphertext consists of a key $H'(Y^x)$ and the group element g^x and a Schnorr signature $(s \coloneqq r + H(g^r, g^x) \cdot x \bmod p, g^r)$ for a random exponent r.

The reduction from [FPS20] ultimately needs to solve a discrete logarithm challenge $dlog_g(Y)$. We choose the basis for the algebraic wrapper as $\{g, Y\}$ in order to enable extraction relative to both g and Y and to allow for (private) sampling of wrapper group elements which depend on Y (whose discrete logarithm must not be known during a reduction to the discrete logarithm problem). For the purpose of reproducing the proof of [FPS20], we first make a conceptual change as in [FPS20] to choose x depending on y (note that at this point of time, y is still known to the experiment). Namely, we use a publicly sampled group element encoding for g^r to query the random oracle $H(g^r, g^x)$ but do not include this encoding in the signature. Instead, we sample a fresh encoding for the group element $g^{s-H(g^r,g^x)\cdot x}$ and use this in place of the encoding of g^r. Since this encoding represents the same group element, the view of the adversary remains unchanged. Then, we apply switching to use the basis $\{g, Y\}$ and sample all encodings using the second decomposition entry instead of explicitly using y in the first decomposition entry. These preparations pave the way to conclude the proof following the lines of [FPS20].

Another technical challenge is to information-theoretically hide whether group elements were sampled directly or derived using the group operation. We achieve this by exploiting the statistical re-randomizability of group elements. This technique is described in this thesis and is required for the tight security proof of Schnorr signatures which is not part of this thesis.

8.2 RELATED WORK

This part builds upon the line of work [AFH+16; AH18; FHHL18; AMP20] who build group schemes from IO. Albrecht et al. [AFH+16] lay the conceptual foundations for the construction of group schemes with non-unique encodings from IO and use this framework to equip groups with multilinear maps. [FHHL18] extend this approach by allowing partial evaluations of the multilinear map yielding a graded encoding scheme. [AMP20] extend [AFH+16; FHHL18] such that computational assumptions in the base group hold, thereby proving equivalence between IO and multilinear maps (modulo subexponential reductions and relatively mild assumptions). In contrast to [AFH+16; FHHL18], the authors of [AH18] do not extend the functionality of an underlying group, but build a group scheme with reduced functionality (group elements lack a unique representation). The resulting group scheme allows to mimic commonly used proof techniques from the generic group model. This is demonstrated by proving the validity of an adaptive variant of the Uber assumption family [Boy08] in the constructed group scheme. Our results can hence be viewed as an extension of [AH18].

[KP19] make a first step towards instantiating the AGM. The authors identify an equivalence between the AGM and a very strong generalized version of the knowledge-of-exponent assumption [Dam92], thus giving rise to the first instantiation of the AGM.

9 PRELIMINARIES

In this chapter, we introduce the necessary preliminaries for this part. Since this part is dedicated to the study of cryptographic groups, we start by introducing additional notations for this purpose (cf. Section 9.1). In Section 9.2 we define the algebraic group model. Subsequently, we introduce useful cryptographic primitives such as subset membership problems in Section 9.3 and dual-mode non-interactive witness indistinguishable proofs in Section 9.4 which are necessary for our construction.

9.1 NOTATIONS AND CRYPTOGRAPHIC GROUPS

Let $\mathbb{K}$ denote a field and let V be a vector space over $\mathbb{K}$ of finite dimension n. For $i \in [n]$, $\boldsymbol{e}_i$ denotes the unit vector, i. e., the vector which carries 1 in its i-th entry and 0 in all other entries. Unless stated otherwise, we work with column vectors.

Let $\mathbb{G}$ denote a finite cyclic group with generator g and order p. For $x \in \mathbb{Z}_p$, the notation $[x]_{\mathbb{G}}$ denotes the group element g^x. Note that using this notation does not imply knowledge of x. The group $\mathbb{G}$ is associated with a group parameter generation algorithm GGen. On input of a security parameter (in unary), GGen samples the parameters $pp_{\mathbb{G}}$ fully describing the group, such that the group operation can be computed in time polynomial in the security parameter. We define hardness assumptions always relative to a group parameter generation algorithm.

DEFINITION 9.1 (Discrete logarithm assumption). The *discrete logarithm (DL) assumption* holds relative to GGen if for all PPT adversaries $\mathcal{A}$,

$$\mathrm{Adv}^{\mathsf{dl}}_{\mathsf{GGen},\mathcal{A}}(\lambda) \coloneqq \Pr\left[\begin{matrix} pp_{\mathbb{G}} \leftarrow \mathsf{GGen}(1^\lambda) \\ x \leftarrow \mathbb{Z}_p \end{matrix} : \mathcal{A}(pp_{\mathbb{G}}, g^x) = x\right]$$

is negligible in λ.

9.2 THE ALGEBRAIC GROUP MODEL

The algebraic group model was introduced in [FKL18] as a model that lies between the generic group model (GGM) [Sho97; Mau05] and the standard model. The algebraic group model is weaker than the generic group model in the sense that in the AGM, an adversary has explicit access to (encodings of) group elements and may very well derive information from them. On the other hand, the algebraic group model still requires that new group elements can only be produced by generic group operations. This is formalized by the ability of the adversary to present a decomposition of all newly produced group elements relative to given group elements.

Formally, [FKL18] only quantify over the set of *algebraic adversaries* (which can explain how they produced new group elements).

DEFINITION 9.2 (Algebraic algorithm (informal), [FKL18]). An algorithm $\mathcal{A}$ is *algebraic* if for all group elements h which $\mathcal{A}$ outputs, it additionally provides a linear decomposition of h relative to all input group elements (only using generic group operations).

[KP19] introduce an equivalent notion of *algebraic groups* where every adversary is bound to be algebraic. By this view of the algebraic group model boils down to an extremely strong knowledge-type assumption since every adversary who outputs a group element *must know* how he computed it. More precisely, [KP19] prove that the AGM is equivalent to an extremely strong generalized version of the knowledge-of-exponent assumption. That is, the AGM (formulated as an assumption[39]) does not enjoy efficient falsifiability due to Naor [Nao03].

[39] *"There exists a group, such that all adversaries in this group are algebraic."*

Further, the AGM as in [FKL18] allows algebraic adversaries to receive "non-group-element inputs" (which are required to be independent of group elements). However, these "non-group-element inputs" to the adversary may contain an obfuscation of a program which on input of a group element $[1]_G$ outputs $[1]_G^{F(K,[1]_G)} = [F(K,[1]_G)]_G$ (where F is a pseudorandom function and K is a random PRF key). Clearly, this program description is independent of group elements (only the input to the program depend on group elements). An adversary who evaluates this obfuscated program on, e. g., the generator of the group and outputs the resulting group element is unable to explain how he computed this group element. In other words, the AGM (defined via adversarial knowledge as in [KP19]) allowing arbitrarily long unstructured "non-group-element inputs" implies extractable one-way functions with unbounded auxiliary information (cf. Definition 2.21) which are known to conflict with indistinguishability obfuscation [BCPR16].

9.3 SUBSET MEMBERSHIP PROBLEM

DEFINITION 9.3 (Subset membership problem, [CS02]). A hard *subset membership problem (SMP)* SMP is defined by an SMP instance generator $\mathsf{Gen}_{\mathsf{SMP}}$.

- $\mathsf{Gen}_{\mathsf{SMP}}$, on input of 1^λ, samples a description of a language and a universe $L \subseteq X$ together with a relation R, such that for all $x \in X$,

$$x \in L \Leftrightarrow \exists! w \in \{0,1\}^{poly(\lambda)} \colon (x,w) \in R.$$

We require that uniform sampling from L (together with the unique witness[40]) and uniform sampling from $X \setminus L$ is efficiently possible. Further, we require that for every PPT adversary $\mathcal{A}$, the advantage

$$\mathrm{Adv}^{\mathsf{smp}}_{\mathsf{Gen}_{\mathsf{SMP}},\mathcal{A}}(\lambda) := \left| \Pr\left[\begin{matrix} (L,X,R) \leftarrow \mathsf{Gen}_{\mathsf{SMP}}(1^\lambda) \\ x \leftarrow L \end{matrix} : \mathcal{A}(1^\lambda, x) = 1 \right] - \Pr\left[\begin{matrix} (L,X,R) \leftarrow \mathsf{Gen}_{\mathsf{SMP}}(1^\lambda) \\ x \leftarrow X \setminus L \end{matrix} : \mathcal{A}(1^\lambda, x) = 1 \right] \right|$$

is negligible in λ.

[40] *Note that unique witness relations are related to the complexity class unambiguous polynomial-time ($\mathcal{UP}$) [Val76].*

Let $\mathbb{G}$ be a cyclic group (of prime order) with group parameter generation algorithm GGen in which DDH is assumed to hold relative to GGen. The Diffie-Hellman language is a hard subset membership problem. The corresponding SMP instance generator $\mathsf{Gen}_{\mathsf{SMP}}$ samples $pp_{\mathbb{G}} \leftarrow \mathsf{GGen}$ and a uniformly random exponent $x \leftarrow \mathbb{Z}_p$, and outputs the language and the universe

$$L_{pp_{\mathbb{G}},[(1,x)]_{\mathbb{G}}} \coloneqq \left\{ [(y, xy)]_{\mathbb{G}} \mid y \in \mathbb{Z}_p \right\} \subseteq \mathbb{G} \times \mathbb{G} =: X_{pp_{\mathbb{G}}}$$

and the relation

$$R_{pp_{\mathbb{G}},x} \coloneqq \left\{ [(a, b)]_{\mathbb{G}} \mid b = x \cdot a \bmod p \right\}.$$

Another instantiation of Definition 9.3 is the language containing all commitments to a fixed value using a perfectly binding non-interactive commitment scheme with unique opening. The universe X is then defined as the set of all valid commitments.

9.4 DUAL-MODE NIWI PROOF SYSTEM

A dual-mode non-interactive witness indistinguishable (NIWI) proof system is a variant of general NIWI proofs [FS90] additionally offering two computationally indistinguishable modes to setup the common reference string (CRS). A CRS generated in binding mode provides perfect soundness guarantees whereas a CRS generated in hiding mode provides perfect witness indistinguishability guarantees.

Note that NIWI proof systems do not inherently require a trusted setup such as a common reference string, e. g., [BP15a]. These instantiations, however, do not satisfy perfect soundness and perfect witness-indistinguishability simultaneously. For our purpose, we require a notion of NIWI proof system which offers two indistinguishable modes of operation: one satisfying perfect soundness and the other satisfying perfect witness-indistinguishability.

DEFINITION 9.4 (Dual-mode NIWI proof system, [GS08; AFH+16]). A *dual mode non-interactive witness-indistinguishable (NIWI) proof system* for a relation $\mathcal{R}$ is a tuple of PPT algorithms $\mathsf{NIWI} = (\mathsf{Setup}, \mathsf{HSetup}, \mathsf{Prove}, \mathsf{Verify}, \mathsf{Ext})$.

- $\mathsf{Setup}(1^\lambda)$ outputs a *perfectly binding* common reference string σ and a corresponding extraction trapdoor td_{ext}.
- $\mathsf{HSetup}(1^\lambda)$ outputs a *perfectly hiding* common reference string σ.
- Prove, on input of a CRS σ, a statement x and a corresponding witness w, produces a proof π.
- Verify, on input of a CRS σ, a statement x and a proof π, deterministically outputs 1 if the proof is valid and 0 otherwise.
- Ext, on input of the extraction trapdoor td_{ext}, a statement x and a proof π, outputs a witness w.

We require NIWI to satisfy the following properties.

CRS INDISTINGUISHABILITY. For all PPT adversaries $\mathcal{A}$,

$$\mathrm{Adv}^{\mathsf{crs\text{-}ind}}_{\mathsf{NIWI},\mathcal{A}}(\lambda) \coloneqq \Big| \Pr\left[(\sigma, td_{ext}) \leftarrow \mathsf{Setup}(1^\lambda) \colon \mathcal{A}(1^\lambda, \sigma) = 1\right] - \Pr\left[\sigma \leftarrow \mathsf{HSetup}(1^\lambda) \colon \mathcal{A}(1^\lambda, \sigma) = 1\right] \Big|$$

is negligible.

PERFECT COMPLETENESS. For all $\lambda \in \mathbb{N}$, all PPT adversaries $\mathcal{A}$, we have that

$$\Pr\left[\begin{array}{rcll}(\sigma,\cdot) & \leftarrow & \mathsf{Setup}(1^\lambda) & \\ (x,w) & \leftarrow & \mathcal{A}(\sigma) & : \\ \pi & \leftarrow & \mathsf{Prove}(\sigma,x,w) & \end{array}\begin{array}{c}(x,w)\notin\mathcal{R}\\ \vee\\ \mathsf{Verify}(\sigma,x,\pi)=1\end{array}\right]=1,$$

where the probability is over the random coins of Setup, Prove and $\mathcal{A}$. The same holds for HSetup. In other words, every proof π produced via Prove for a valid statement-witness-pair will be accepted by Verify.

PERFECT SOUNDNESS IN BINDING MODE. For all $\lambda \in \mathbb{N}$ and all PPT adversaries $\mathcal{A}$, we have that

$$\Pr\left[\begin{array}{rcl}(\sigma,\cdot) & \leftarrow & \mathsf{Setup}(1^\lambda)\\ (x,\pi) & \leftarrow & \mathcal{A}(\sigma)\end{array} : x\in L \wedge \mathsf{Verify}(\sigma,x,\pi)=1\right]=0,$$

where the probability is over the random coins of Setup and $\mathcal{A}$. In other words, all statements $x \notin L$, there exists no proof such that Verify accepts.

PERFECT EXTRACTABILITY IN BINDING MODE. For all $\lambda \in \mathbb{N}$ and all PPT adversaries $\mathcal{A}$, we have

$$\Pr\left[\begin{array}{rcl}(\sigma,td_{ext}) & \leftarrow & \mathsf{Setup}(1^\lambda)\\ (x,\pi) & \leftarrow & \mathcal{A}(\sigma)\end{array} : \begin{array}{c}\mathsf{Verify}(\sigma,x,\pi)=0\\ \vee\\ \big(x,\mathsf{Ext}(td_{ext},x,\pi)\big)\in\mathcal{R}\end{array}\right]=1,$$

where the probability is over the random coins of Setup, $\mathcal{A}$ and Ext. In other words, from every accepting statement-proof-pair (x,π), the extraction algorithm Ext extracts a witness w such that $(x,w) \in \mathcal{R}$.

PERFECT WITNESS-INDISTINGUISHABILITY IN HIDING MODE. For every $\lambda \in \mathbb{N}$, all $\sigma \in \mathit{supp}(\mathsf{HSetup}(1^\lambda))$, all $(x,w_0),(x,w_1) \in \mathcal{R}$ (of arbitrary but polynomial length in λ), the output distributions induced by

$$\mathsf{Prove}(\sigma,x,w_0) \quad \text{and} \quad \mathsf{Prove}(\sigma,x,w_1)$$

are identical.

There are several instantiations of dual-mode NIWI proof systems satisfying the above definition. The construction [GS08] relies on pairing-friendly groups where either the pairing is asymmetric and the SXDH assumption holds, or the pairing is symmetric and the DLIN assumption holds. [PS19] build a (statistically secure, statistically extractable and statistically zero-knowledge) dual-mode NIZK from the learning with errors assumption. The recent paper [HU19] proposes an instantiation of (a statistically secure, statistically extractable and statistically witness-indistinguishable) dual-mode NIWI proof system based on strong but mostly unstructured assumptions, namely, subexponentially secure indistinguishability obfuscation, subexponentially secure one-way functions and a secure lossy encryption scheme. This instantiation is sufficient for our purpose, but, for simplicity, we work with dual-mode NIWI proof systems providing perfect guarantees in our proofs in Chapter 12.

9.5 RE-RANDOMIZABLE PUBLIC-KEY ENCRYPTION

A re-randomizable public-key encryption scheme allows to perfectly re-randomize any ciphertext. This can be a useful property in scenarios, where it is necessary to *information-theoretically* hide the randomness which was used for ciphertext generation.

DEFINITION 9.5 (Perfectly re-randomizable public-key encryption, [PR07]). A *perfectly re-randomizable public-key encryption scheme* with message space $\{0,1\}^*$ is a tuple of PPT algorithms $\mathsf{E} = (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Rerand})$ such that $(\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ is a perfectly correct PKE scheme such that the following additional properties are met.

- For all $(pk, \cdot) \in \mathit{supp}(\mathsf{KGen}(1^\lambda))$, all messages $m \in \{0,1\}^*$, all ciphertexts $C \in \mathit{supp}(\mathsf{Enc}(pk, m))$, $\mathsf{Rerand}(C)$ is distributed identically to $\mathsf{Enc}(pk, m)$.
- For all $(pk, \cdot) \in \mathit{supp}(\mathsf{KGen}(1^\lambda))$, all (maliciously chosen) ciphertexts C, and all $C' \in \mathit{supp}(\mathsf{Rerand}(C))$, $\mathsf{Dec}(sk, C') = \mathsf{Dec}(sk, C)$.

For our purposes it suffices to let E.Rerand receive the public key as input. Furthermore, in contrast to [PR07], we do not require that with overwhelming probability over the choice of $(pk', \cdot) \leftarrow \mathsf{E.KGen}(1^\lambda)$, the ciphertext spaces under pk and pk' are disjoint, i. e., $\mathit{supp}(\mathsf{E.Enc}(pk, \cdot)) \cap \mathit{supp}(\mathsf{E.Enc}(pk', \cdot)) = \varnothing$. The ElGamal encryption scheme [ElG85], the Goldwasser-Micali encryption scheme [GM84], the Paillier encryption scheme [Pai99] and the Damgård-Jurik encryption scheme [DJ01] are perfectly re-randomizable public-key encryption schemes, [PR07; HLOV11].

It is currently unclear, whether lossy PKE schemes (and hence re-randomizable PKE schemes, [HLOV11][41]) can be built from IO in conjunction with unstructured generic assumptions like OWFs.

Note that the FHE construction due to Canetti et al. [CLTV15] as well as our FHE construction from Part I preserve re-randomizability of the underlying PKE scheme. That is, any perfectly re-randomizable PKE scheme can be converted into a perfectly re-randomizable FHE scheme. Such a FHE scheme will be necessary for our construction in Chapter 12.

[41] *[HLOV11] build a lossy encryption scheme from a re-randomizable encryption scheme, where an "injective" public key contains a ciphertext C_0 of 0 and a ciphertext C_1 of 1 and a "lossy" public key contains two ciphertexts C_0, C_1 of 0. Encryption of a bit b simply re-randomizes the ciphertext C_b and decryption works as in the underlying scheme.*

10 STATISTICALLY CORRECT PIO

Probabilistic IO allows to obfuscate a randomized computation such that the user of the obfuscated code is unable to control the used randomness. Moreover, the randomness used by the obfuscated circuit is *pseudorandom* from the view of the circuit user – even though he possesses the code. Hence, probabilistic IO can be used to obfuscate a randomized circuit which produces a ciphertext of some plaintext which should remain hidden from the circuit user. In other words, pIO enables to protect the randomness of some probabilistic process from its user. Consequently, pIO fills an important gap which is left open by (deterministic) indistinguishability obfuscation.

On the other hand, it may be undesirable to leak information on *how* some output was computed. In particular, one may be interested to hide whether the obfuscated circuit or the original randomized circuit (with fresh randomness) was used. Probabilistic IO hides this information against computationally bounded observers. However, since the obfuscated program is deterministic, it is information-theoretically evident whether the original randomized circuit or the obfuscated (deterministic) circuit was used. This is an inherent limitation of probabilistic IO. However, there are scenarios, where the origin of some output is required to be *information-theoretically* hidden.[42] A natural question that arises is whether this is feasible.

[42] *One such scenario is described and studied later in Chapter 12.*

In the following, we define a notion of *statistically correct* probabilistic IO and provide an instantiation of this notion. More precisely, we require statistical closeness between evaluations of the original (probabilistic) circuit and the obfuscated (deterministic) circuit. By definition of probabilistic IO, this is impossible since the obfuscated circuit is deterministic and, hence, has no source of entropy other than its input. However, we observe that as long as a portion of the circuit's input is guaranteed to be outside the view of the adversary (and has sufficiently high min-entropy), the output of the obfuscated circuit and the actual probabilistic circuit can be statistically close.

For this purpose, we extend the definition of probabilistic IO as follows. Before obfuscating a probabilistic circuit, we compile this circuit such that it receives an "auxiliary input" *aux* but simply ignores this input in its computation. Even though the obfuscated circuit is deterministic, this auxiliary input can be used as a source of *actual entropy*.

FIRST TRY. We recall that the pIO construction from [CLTV15] obfuscates a probabilistic circuit C by using IO to obfuscate the deterministic circuit $\overline{C}(x) := C(x; F_K(x))$. A natural idea to achieve statistical correctness is to modify this construction such that the auxiliary input *aux* is directly XORed on the random tape which is derived using F, i.e. to obfuscate the circuit $\overline{C}(x, aux; F_K(x) \oplus aux)$. For uniform auxiliary input *aux*, statistical correctness follows immediately. However, security breaks down. Consider two circuits C_1 and C_2 such that C_1 outputs the first bit on its random tape and C_2 outputs the second bit on its random tape. Since C_1 and C_2 produce identical output distributions, it is desirable that a probabilistic indistinguishability

obfuscator conceals which of the two circuits was obfuscated. However, this construction admits a successful attack. An adversary can evaluate the obfuscated circuit Λ on inputs (x, aux) and $(x, aux \oplus 1)$. If both evaluations yield identical outputs, C_2 was obfuscated, otherwise C_1 was obfuscated.

USING AN EXTRACTING PRF. Our construction of statistically correct pIO applies an extracting puncturable PRF on the entire input of the circuit to derive the random tape for the probabilistic circuit. An extracting PRF guarantees that PRF outputs are uniformly distributed (even given the PRF key) as long as the input has high min-entropy. This is achieved using a universal hash function and the leftover hash lemma.

OUTLINE. In Section 10.1, we formally define statistically correct pIO. In Section 10.2 we introduce preliminaries which are necessary for our instantiation. Finally, in Section 10.3, we provide an instantiation of statistically correct pIO.

10.1 STATISTICALLY CORRECT PIO

In this section, we formally define our novel notion of statistically correct input-expanding pIO. We start by defining expanding compilers and expanding circuit samplers.

DEFINITION 10.1 (ℓ-expanding compiler). An *ℓ-expanding compiler* $\mathcal{E}_\ell$ takes as input a probabilistic circuit C of size $p'(\lambda)$, expecting inputs $x \in \{0,1\}^{n'(\lambda)}$ and randomness $r \in \{0,1\}^{m(\lambda)}$, and outputs a circuit $\widehat{C}$ of polynomial size $p(\lambda) = p'(\lambda) + \ell(\lambda)$ expecting inputs $(x, aux) \in \{0,1\}^{n'(\lambda)} \times \{0,1\}^{\ell(\lambda)}$, randomness $r \in \{0,1\}^{m(\lambda)}$ such that for all $x \in \{0,1\}^{n'(\lambda)}$, all $aux \in \{0,1\}^{\ell(\lambda)}$ and all $r \in \{0,1\}^{m(\lambda)}$,

$$C(x; r) = \widehat{C}(x, aux; r).$$

The compiler $\mathcal{E}_\ell$ which simply appends $\ell(\lambda)$ input gates (without any additional edges) to the original circuit satisfies the above definition.

Our construction of pIO will first expand the given circuit as above and then use a slightly modified version of the pIO scheme of [CLTV15] to obfuscate the expanded circuit. In the following, we formally define the properties of this expanding probabilistic indistinguishability obfuscator. Let $\mathcal{C} = (\mathcal{C}_\lambda)_{\lambda \in \mathbb{N}}$ be an ensemble of sets $\mathcal{C}_\lambda$ of probabilistic circuits. The set $\mathcal{C}_\lambda$ contains circuits of size λ with input length $n'(\lambda)$ expecting (at most) $m(\lambda)$ random bits. Let $\widehat{\mathcal{C}} = \{\widehat{\mathcal{C}}_\lambda\}_{\lambda \in \mathbb{N}}$ be the circuit class which results from ℓ-expanding the circuit class $\mathcal{C}$. That is, $\widehat{\mathcal{C}}_{\lambda+\ell(\lambda)} = \mathcal{E}_\ell(\mathcal{C}_\lambda)$.[43]

Let D be a circuit sampler for the circuit class $\mathcal{C}$. Let $\widehat{D} := \{\widehat{D}_\lambda\}_{\lambda \in \mathbb{N}}$ be the corresponding ℓ-expanding circuit sampler. More formally, $\widehat{D}_{\lambda+\ell(\lambda)}$ samples (C_0, C_1, z) from the distribution D_λ and outputs the circuits $\widehat{C}_0 := \mathcal{E}_\ell(C_0)$, $\widehat{C}_1 := \mathcal{E}_\ell(C_1)$ and auxiliary information $\widehat{z} := (C_0, C_1, z)$.[44] The circuit sampler $\widehat{D}$ is a circuit sampler for the circuit class $\widehat{\mathcal{C}}$.

By the security of pIO for the class of $\mathfrak{C}^{\text{X-ind}}$ due to [CLTV15], our input-expanding pIO scheme is secure for the following class of input-expanding X-ind samplers.

[43] *Note that this is well-defined since we implicitly fix the function $\ell(\lambda)$ and require that $\ell(\lambda)$ is strictly increasing. This is without loss of generality.*

[44] *Note that since the compilation process of $\mathcal{E}_\ell$ is reversible, we could actually omit C_0, C_1 from $\widehat{z}$.*

DEFINITION 10.2 (ℓ-expanding X-ind sampler). *An ℓ-expanding X-ind sampler* D for the circuit class $\mathcal{C}$ is a circuit sampler for $\mathcal{C}$ such that its corresponding ℓ-expanding circuit sampler $\widehat{\mathsf{D}}$ is an X-ind sampler according to Definition 2.18. We denote the class of all ℓ-expanding X-ind samplers $\mathfrak{C}_\ell^{\mathsf{X}\text{-}(\star)\mathsf{ind}}$.

Unfortunately, not all X-ind samplers D (for $\mathcal{C}$) induce a circuit sampler $\widehat{\mathsf{D}}$ (for $\widehat{\mathcal{C}}$) which is an $\widehat{\mathsf{X}}$-ind sampler. In order for $\widehat{\mathsf{D}}$ to satisfy the $\widehat{\mathsf{X}}$-differing inputs property, we need to set the function $\widehat{\mathsf{X}}$ such that $\widehat{\mathsf{X}}(\lambda+\ell(\lambda)) = \mathsf{X}(\lambda)\cdot 2^{\ell(\lambda)} \leqslant 2^{\lambda+\ell(\lambda)}$, where λ is the security parameter used for D and $\lambda+\ell(\lambda)$ is the security parameter for $\widehat{\mathsf{D}}$.[45] $\widehat{\mathsf{X}}$-indistinguishability, however, requires that

$$\widehat{\mathsf{X}}(\lambda+\ell(\lambda))\cdot \mathrm{Adv}^{\mathsf{sel\text{-}ind}}_{\widehat{\mathsf{D}},\widehat{\mathcal{A}}}(\lambda+\ell(\lambda)) = \mathsf{X}(\lambda)\cdot 2^{\ell(\lambda)}\cdot \mathrm{Adv}^{\mathsf{sel\text{-}ind}}_{\widehat{\mathsf{D}},\widehat{\mathcal{A}}}(\lambda+\ell(\lambda))$$

is negligible (for all PPT adversaries $\widehat{\mathcal{A}}$). Due to X-indistinguishability of D, we are only guaranteed that $\mathsf{X}(\lambda)\cdot \mathrm{Adv}^{\mathsf{sel\text{-}ind}}_{\mathsf{D},\mathcal{A}}(\lambda)$ is negligible (for all PPT adversaries $\mathcal{A}$), which does not suffice to prove $\widehat{\mathsf{X}}$-indistinguishability of $\widehat{\mathsf{D}}$.

[45] *Note that $\widehat{\mathsf{D}}$ is called with security parameter $\lambda+\ell(\lambda)$ to compensate for the expanded circuit size.*

Nevertheless, ℓ-expanding X-ind samplers cover a wide class of circuit samplers. In the following we present a lemma which facilitates the usage of ℓ-expanding X-ind samplers.

LEMMA 10.1. *Let D be a circuit sampler for $\mathcal{C}$ which outputs two circuits C_0 and C_1 such that for all inputs x, $\mathsf{C}_0(x)$ and $\mathsf{C}_1(x)$ produce the exact same distribution. Then, $\mathsf{D}\in\mathfrak{C}_\ell^{\mathsf{X}\text{-}(\star)\mathsf{ind}}$.*

Proof. Let $\widehat{\mathsf{D}}$ be the ℓ-expanding circuit sampler corresponding to D. Let $\widehat{\mathsf{X}}(\lambda+\ell(\lambda)) := 2^{\lambda+\ell(\lambda)}$ and $\widehat{\mathcal{X}} := \{0,1\}^{\lambda+\ell(\lambda)}$. Since there is no input for the circuits which is outside of $\widehat{\mathcal{X}}$, the $\widehat{\mathsf{X}}$-differing inputs property is trivially satisfied. Furthermore, as for all inputs x the distributions $\mathsf{C}_0(x)$ and $\mathsf{C}_1(x)$ are identical, the same holds for the distributions $\mathcal{E}_\ell(\mathsf{C}_0)(x,aux)$ and $\mathcal{E}_\ell(\mathsf{C}_1)(x,aux)$ for all inputs x and all $aux\in\{0,1\}^{\ell(\lambda)}$. Thus, for all adversaries $\mathcal{A}$, $\mathrm{Exp}^{\mathsf{sel\text{-}ind}}_{\widehat{\mathsf{D}},\mathcal{A}}(\lambda+\ell(\lambda)) = \frac{1}{2}$. Hence, $\widehat{\mathsf{D}}\in\mathfrak{C}^{\mathsf{X}\text{-}\mathsf{ind}}$ and, therefore, $\mathsf{D}\in\mathfrak{C}_\ell^{\mathsf{X}\text{-}(\star)\mathsf{ind}}$. □

Lemma 10.1 directly yields the following corollary.

COROLLARY 10.1. *Let D be a circuit sampler for $\mathcal{C}$ which outputs two circuits C_0 and C_1 which always behave exactly identically on identical inputs and random tapes. Then, $\mathsf{D}\in\mathfrak{C}_\ell^{\mathsf{X}\text{-}(\star)\mathsf{ind}}$.*

Our expanding pIO scheme satisfies similar correctness and security properties as defined in [CLTV15] but additionally guarantees statistical correctness.

Let $\mathcal{C}=(\mathcal{C}_\lambda)_{\lambda\in\mathbb{N}}$ be an ensemble of sets $\mathcal{C}_\lambda$ of probabilistic circuits. The set $\mathcal{C}_\lambda$ contains circuits of size λ with input length $n'(\lambda)$ expecting (at most) $m(\lambda)$ random bits. Further, let $\mathfrak{C}$ be a class of circuit samplers over $\mathcal{C}$.

DEFINITION 10.3 (ℓ-expanding pIO for the class of samplers $\mathfrak{C}$). *An ℓ-expanding probabilistic indistinguishability obfuscator* for the class of samplers $\mathfrak{C}$ over $\mathcal{C}$ is a uniform PPT algorithm $\mathsf{piO}^\star_\ell$, satisfying the following properties.

INPUT EXPANDING CORRECTNESS. For all PPT adversaries $\mathcal{A}$ and all circuits $\mathsf{C}\in\mathcal{C}$,

$$\Big|\Pr\left[\mathcal{A}^{\mathcal{O}_{\mathsf{C}}(\cdot,\cdot)}(1^\lambda,\mathsf{C})=1\right] - \Pr\left[\Lambda\leftarrow\mathsf{piO}^\star_\ell(1^\lambda,\mathsf{C}):\mathcal{A}^{\mathcal{O}_\Lambda(\cdot,\cdot)}(1^\lambda,\mathsf{C})=1\right]\Big|$$

is negligible as a function in λ, where the oracles $\mathcal{O}_C$ and $\mathcal{O}_\Lambda$ are defined in Figure 10.1 and must not be called twice on the same input (x, aux).

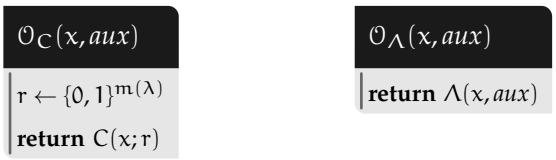


Figure 10.1: The oracles used in the definition of correctness of ℓ-expanding pIO.

SECURITY WITH RESPECT TO $\mathfrak{C}$. For all circuit samplers $D \in \mathfrak{C}$, for all PPT adversaries $\mathcal{A}$, the advantage

$$\begin{aligned}\mathrm{Adv}^{\text{pio-ind}(\star)}_{\mathsf{piO}^\star_\ell, D, \mathcal{A}}(\lambda) \coloneqq \\ \Big| \Pr\left[(C_0, C_1, z) \leftarrow D(1^\lambda) \colon \mathcal{A}(C_0, C_1, z, \mathsf{piO}^\star_\ell(1^\lambda, C_0)) = 1\right] \\ - \Pr\left[(C_0, C_1, z) \leftarrow D(1^\lambda) \colon \mathcal{A}(C_0, C_1, z, \mathsf{piO}^\star_\ell(1^\lambda, C_1)) = 1\right] \Big|\end{aligned}$$

is negligible in λ.

SUPPORT RESPECTING. Let $n'(\lambda)$ denote the input length of circuits in $\mathcal{C}_\lambda$. For all circuits $C \in \mathcal{C}_\lambda$, all inputs $x \in \{0,1\}^{n'(\lambda)}$, all $aux \in \{0,1\}^{\ell(\lambda)}$, all $\Lambda \in \mathit{supp}(\mathsf{piO}^\star_\ell(1^\lambda, C))$, we have that

$$\Lambda(x, aux) \in \mathit{supp}(C(x)).$$

STATISTICAL CORRECTNESS WITH ERROR $2^{-e(\lambda)}$. For all circuits $C \in \mathcal{C}_\lambda$ and all joint distributions (Z_1, Z_2) over $\{0,1\}^{n'(\lambda)} \times \{0,1\}^{\ell(\lambda)}$ with average min-entropy $\ell(\lambda) \geqslant \widetilde{H}_\infty(Z_2 \mid Z_1) > m(\lambda) + 2e(\lambda) + 2$, the statistical distance

$$\Delta\Big(\left\{\Lambda \leftarrow \mathsf{piO}^\star_\ell(1^\lambda, C) \colon (\Lambda, \Lambda(Z_1, Z_2))\right\}, \\ \left\{\Lambda \leftarrow \mathsf{piO}^\star_\ell(1^\lambda, C) \colon (\Lambda, C(Z_1; U_{m(\lambda)}))\right\}\Big)$$

is at most $2^{-e(\lambda)}$.

Setting $\ell \coloneqq 0$ recovers the original definition of pIO due to [CLTV15], see Definition 2.16. Furthermore, instantiating an easy modification of the construction of piO for $\mathfrak{C}^{\text{X-ind}}$ due to [CLTV15] with a suitably extracting PRF family satisfies our definition of ℓ-expanding pIO.

10.2 PUNCTURABLE PSEUDORANDOM FUNCTIONS

As preparation for our construction, we introduce several variants of puncturable pseudorandom functions (pPRFs). Puncturable PRFs allow to puncture a key at a certain amount of inputs, such that the punctured key works on all remaining inputs as before. Furthermore, evaluations at punctured points are pseudorandom even given the punctured key. We refer the reader to Definition 2.11 in Section 2.4 for more details.

DEFINITION 10.4 (Statistically injective puncturable PRF, [SW14]). A *statistically injective* puncturable PRF family with error $\epsilon(\cdot)$ is a family of puncturable PRFs F mapping $n(\lambda)$ bits to $m(\lambda)$ bits such that $\mathsf{F}(K,\cdot)$ is injective with probability at least $1-\epsilon(\lambda)$.

Such PRFs are known to exist from one-way functions [SW14].

LEMMA 10.2 ([SW14]). *If one-way functions exist, then for all efficiently computable functions $n(\cdot), m(\cdot), e(\cdot)$ such that $m(\lambda) \geqslant 2n(\lambda) + e(\lambda)$, there exists a statistically injective puncturable PRF family with error $2^{-e(\lambda)}$ mapping $n(\lambda)$ bits to $m(\lambda)$ bits.*

As already indicated, we instantiate the construction of [CLTV15] with a pPRF which allows to use the entropy provided via the auxiliary input *aux* such that the obfuscated circuit is evaluated with statistically uniform random coins. The following definition which extends the definition of extracting PRFs (cf. Definition 2.12 in Section 2.4) captures this property.

DEFINITION 10.5 (Special extracting puncturable PRF, [SW14]). A *special extracting* puncturable PRF family with error $\epsilon(\cdot)$ for average min-entropy $k(\cdot)$ is a family of puncturable PRFs F mapping $n(\lambda) = n'(\lambda) + n''(\lambda)$ bits to $m(\lambda)$ bits such that for all λ and all joint distributions (Z_1, Z_2) over $\{0,1\}^{n'(\lambda)} \times \{0,1\}^{n''(\lambda)}$ with average min-entropy $\widetilde{H}_\infty(Z_2 \mid Z_1) > k(\lambda)$, the statistical distance

$$\Delta\Big(\big\{K \leftarrow \mathsf{KGen}(1^\lambda)\colon \big(K, Z_1, \mathsf{F}(K,Z)\big)\big\},\\ \big\{K \leftarrow \mathsf{KGen}(1^\lambda)\colon \big(K, Z_1, U_{m(\lambda)}\big)\big\}\Big) \leqslant \epsilon(\lambda),$$

where the distribution Z is defined as $Z \coloneqq (Z_1, Z_2)$.

Note that extracting PRFs (cf. Definition 2.12 in Section 2.4) do not suffice to deal with joint distributions (Z_1, Z_2), where only a guarantee on the average min-entropy $\widetilde{H}_\infty(Z_2 \mid Z_1)$ is known. However, this is required by Definition 10.3.

The leftover hash lemma states that universal hash functions are good randomness extractors, cf. Definition 2.4 in Section 2.1. In other words, if the input of a universal hash function has sufficiently high (average) min-entropy, the output of that hash function is statistically close to uniform even given the function description.

LEMMA 10.3 (Leftover Hash Lemma for average min-entropy, [HILL99]). *Let $\mathcal{H}$ be a 2-universal hash function family mapping n to m bits. If $\widetilde{H}_\infty(Z \mid E) \geqslant k$ and $m = k - 2\log(\frac{1}{2\epsilon})$, then*

$$\Delta\Big(\big\{H \leftarrow \mathcal{H}\colon \big(H, E, H(Z)\big)\big\}, \big\{H \leftarrow \mathcal{H}\colon \big(H, E, U_m\big)\big\}\Big) \leqslant \epsilon.$$

For our purpose, we require that the statistical distance between

$$\big\{H \leftarrow \mathcal{H}\colon \big(H, E, H(Z,E)\big)\big\} \quad \text{and} \quad \big\{H \leftarrow \mathcal{H}\colon \big(H, E, U_m\big)\big\}$$

is at most ϵ. Since $\widetilde{H}_\infty(Z \mid E) = \widetilde{H}_\infty((Z,E) \mid E)$, this is implied by Lemma 10.3.

Now we are prepared to extend Theorem 3 in the full version of [SW14] to support joint distributions.

THEOREM 10.1. *If one-way functions exist, then for all efficiently computable functions $n'(\cdot), n''(\cdot), m(\cdot), k(\cdot)$ and $e(\cdot)$ such that $n(\lambda) = n'(\lambda) + n''(\lambda) \geqslant k(\lambda) \geqslant m(\lambda) + 2e(\lambda) + 2$, there exists a special extracting puncturable PRF family mapping $n(\lambda)$ bits to $m(\lambda)$ bits with error $2^{-e(\lambda)}$ for average min-entropy $k(\lambda)$.*

Proof. The proof is very similar to the proof of Theorem 3 in the full version of [SW14]. Let F be a family of statistically injective puncturable PRFs with error $2^{-(e(\lambda)+1)}$ mapping $n(\lambda)$ bits to $2n(\lambda) + e(\lambda) + 1$ bits. By Lemma 10.2, such a PRF family exists from one-way functions. Let $\mathcal{H}$ be a family of 2-universal hash functions mapping $2n(\lambda) + e(\lambda) + 1$ bits to $m(\lambda)$ bits. In Figure 10.2, we define a family $\mathsf{F}' = (\mathsf{KGen}', \mathsf{Eval}', \mathsf{Punct}')$ of puncturable PRFs.

$\mathsf{F}'.\mathsf{KGen}'(1^\lambda)$

$h \leftarrow \mathcal{H}$

$K \leftarrow \mathsf{F}.\mathsf{KGen}(1^\lambda)$

return $K' := (K, h)$

(a) The key generation algorithm KGen'.

$\mathsf{F}'.\mathsf{Eval}'(K', x)$

parse $K' =: (K, h)$

return $h\big(\mathsf{F}.\mathsf{Eval}(K, x)\big)$

(b) The evaluation algorithm Eval'.

$\mathsf{F}'.\mathsf{Punct}'(K', T)$

parse $K' =: (K, h)$

$K\{T\} \leftarrow \mathsf{F}.\mathsf{Punct}(K, T)$

return $K'\{T\} := (K\{T\}, h)$

(c) The key puncturing algorithm Punct'.

Figure 10.2: The definition of our special extracting puncturable PRF F'.

Due to [SW14], F' is a family of puncturable PRFs mapping $n(\lambda)$ bits to $m(\lambda)$ bits. Let (Z_1, Z_2) be a joint distribution over $\{0,1\}^{n(\lambda)} = \{0,1\}^{n'(\lambda)} \times \{0,1\}^{n''(\lambda)}$ such that $\widetilde{H}_\infty(Z_2 \mid Z_1) \geqslant m(\lambda) + 2e(\lambda) + 2$. Fix a key K such that $\mathsf{F}(K, \cdot)$ is injective. Hence, $\widetilde{H}_\infty(Z_2 \mid Z_1) = \widetilde{H}_\infty(\mathsf{F}(K, (Z_1, Z_2)) \mid Z_1)$. Lemma 10.3 implies that the statistical distance

$$\Delta\Big(\big\{h \leftarrow \mathcal{H}\colon \big(h, Z_1, h\big(\mathsf{F}(K, (Z_1, Z_2))\big)\big)\big\}, \big\{h \leftarrow \mathcal{H}\colon \big(h, Z_1, U_{m(\lambda)}\big)\big\}\Big)$$

is at most $2^{-(e(\lambda)+1)}$. The probability that a randomly sampled key K yields a non-injective PRF is at most $2^{-(e(\lambda)+1)}$. Therefore, the statistical distance

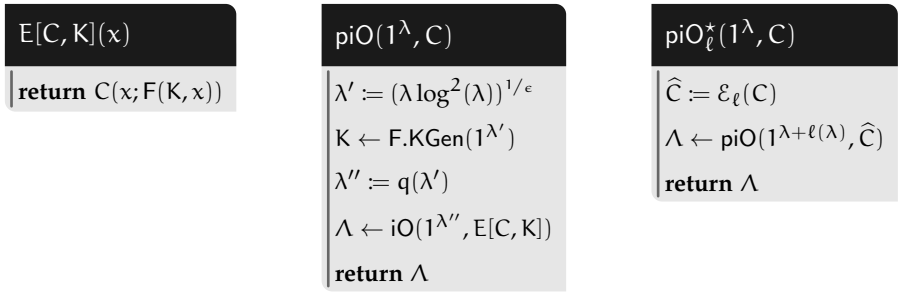
$$\Delta\Big(\big\{K' \leftarrow \mathsf{F}'.\mathsf{KGen}(1^\lambda)\colon \big(K', Z_1, \mathsf{F}'(K', (Z_1, Z_2))\big)\big\}, \big\{K' \leftarrow \mathsf{F}'.\mathsf{KGen}(1^\lambda)\colon \big(K', Z_1, U_{m(\lambda)}\big)\big\}\Big) \leqslant 2^{-e(\lambda)}$$

concluding the proof. □

10.3 CONSTRUCTION

Figure 10.3 defines our construction of an ℓ-expanding pIO scheme $\mathsf{piO}^\star_\ell$. We use the pIO scheme for X-ind samplers piO from [CLTV15] instantiated with a special extracting pPRF as a subroutine.

THEOREM 10.2. *Let $e(\lambda)$ be an efficiently computable function. Let F be a subexponentially secure special extracting PRF family with distinguishing advantage $2^{-\lambda^\epsilon}$ (for some constant ϵ) and error $2^{-e(\lambda)}$ mapping $n(\lambda) := n'(\lambda) + \ell(\lambda)$ bits to $m(\lambda)$ bits which is extracting if the input average min-entropy is greater than*



(a) The circuit $\mathsf{E}[\mathsf{C},\mathsf{K}]$ used in [CLTV15].

(b) The construction of pIO for X-ind samplers from [CLTV15].

(c) Our construction of ℓ-expanding pIO.

Figure 10.3: Construction of pIO for X-ind samplers from [CLTV15] (Figures 10.3a and 10.3b) and of ℓ-expanding pIO (Figure 10.3c). Both definitions are for circuits of size λ. In Figure 10.3b, the polynomial $q(\lambda')$ denotes an upper bound on the size of the circuit $\mathsf{E}[\mathsf{C},\mathsf{K}]$ and F denotes a special extracting subexponentially secure pPRF with distinguishing gap $2^{-\lambda^\epsilon}$ (for a constant ϵ).

$m(\lambda)+2e(\lambda)+2$. *Let* piO *denote the construction of pIO from [CLTV15] instantiated with* F *(cf. Figure 10.3b). Then,* $\mathsf{piO}^\star_\ell$ *as defined in Figure 10.3c is a statistically correct input-expanding pIO for the class of samplers* $\mathfrak{C}_\ell^{\text{X-}(\star)\text{ind}}$.

Proof. We recall that due to [CLTV15], piO satisfies correctness and security with respect to X-ind samplers. We now prove that $\mathsf{piO}^\star_\ell$ defined in Figure 10.3 is a statistically correct input expanding pIO scheme for the class of samplers $\mathfrak{C}_\ell^{\text{X-}(\star)\text{ind}}$.

INPUT EXPANDING CORRECTNESS. Note that input expanding correctness does not follow in a black-box manner from the correctness of piO since input expanding correctness allows an adversary to query the randomized circuit multiple times on the same input x using different auxiliary inputs *aux*. This cannot be simulated by an adversary in the correctness game of piO, cf. Definition 2.16.

Input expanding correctness follows from pseudorandomness of the PRF used in the construction of piO and correctness of iO. Let C be a circuit in $\mathcal{C}_\lambda$. Let $\mathcal{A}$ be a PPT adversary. Let $\mathbf{G}_0$ denote the game in which the adversary gets C as input and has oracle access to $\mathcal{O}_C(\cdot,\cdot)$ and let $\mathbf{G}_2$ denote the game in which the adversary gets C as input and has oracle access to $\mathcal{O}_\Lambda(\cdot,\cdot)$ for $\Lambda \leftarrow \mathsf{piO}^\star_\ell(1^\lambda, C)$. Both oracles must not be called twice on the same input. Consider an intermediate game $\mathbf{G}_1$, where the oracle $\mathcal{O}_\Lambda(x, aux)$ evaluates the circuit $\widehat{C}(x, aux; \mathcal{R}(x, aux))$ for a truly random function $\mathcal{R} \in \mathit{maps}(\{0,1\}^{n'(\lambda)+\ell(\lambda)}, \{0,1\}^{m(\lambda)})$. By the security of the PRF F, oracle access to $\mathsf{F}(\mathsf{K},\cdot)$ (for a random key K) and a truly random function are indistinguishable. Hence, by perfect correctness of iO and the security of F, $\mathbf{G}_1$ and $\mathbf{G}_2$ are indistinguishable. Furthermore, since $\widehat{C}(x, aux; \mathcal{R}(x, aux)) = C(x; \mathcal{R}(x, aux))$ and by perfect correctness of iO, $\mathbf{G}_0$ and $\mathbf{G}_1$ are identically distributed.

SECURITY WITH RESPECT TO $\mathfrak{C}^{\text{X-ind}}$. Since piO is secure with respect to X-ind samplers (and $\mathcal{E}_\ell(C) \in \widehat{\mathcal{C}}$), $\mathsf{piO}^\star_\ell$ is secure with respect to ℓ-expanding X-ind samplers.

More formally, let $\mathsf{D} \in \mathfrak{C}_\ell^{\text{X-}(\star)\text{ind}}$ be a circuit sampler and let $\mathcal{A}$ be a PPT distinguisher for $\mathsf{piO}_\ell^\star$-obfuscations of C_0 and C_1 as sampled from D, cf. Definition 10.3. We construct a circuit sampler $\widehat{\mathsf{D}}$ (as described above) and an adversary $\mathcal{B}$ for the security property of piO. Let $\widehat{\mathsf{D}}$ be the ℓ-expanding circuit sampler corresponding to D. That is, $\widehat{\mathsf{D}}_{\lambda+\ell(\lambda)}$, samples from D_λ to obtain $(\mathsf{C}_0, \mathsf{C}_1, z)$ and outputs $(\mathcal{E}_\ell(\mathsf{C}_0), \mathcal{E}_\ell(\mathsf{C}_1), \widehat{z} \coloneqq (z, \mathsf{C}_0, \mathsf{C}_1))$. On input of $(\mathcal{E}_\ell(\mathsf{C}_0), \mathcal{E}_\ell(\mathsf{C}_1), \widehat{z}, \Lambda)$, $\mathcal{B}$ calls $\mathcal{A}$ on input of $(\mathsf{C}_0, \mathsf{C}_1, z, \Lambda)$ and outputs $\mathcal{A}$'s output. Hence, for $\mathsf{b} \in \{0,1\}$, we have

$$\begin{aligned}
&\Pr\Big[(\mathsf{C}_0, \mathsf{C}_1, z) \leftarrow \mathsf{D}(1^\lambda) \colon \mathcal{A}\big(\mathsf{C}_0, \mathsf{C}_1, z, \mathsf{piO}_\ell^\star(1^\lambda, \mathsf{C}_\mathsf{b})\big) = 1\Big] \\
=&\Pr\Big[\big(\mathcal{E}_\ell(\mathsf{C}_0), \mathcal{E}_\ell(\mathsf{C}_1), (\mathsf{C}_0, \mathsf{C}_1, z)\big) \leftarrow \widehat{\mathsf{D}}(1^{\lambda+\ell(\lambda)}) \colon \\
&\qquad \mathcal{B}\big(\mathcal{E}_\ell(\mathsf{C}_0), \mathcal{E}_\ell(\mathsf{C}_1), (\mathsf{C}_0, \mathsf{C}_1, z), \mathsf{piO}\big(1^{\lambda+\ell(\lambda)}, \mathcal{E}_\ell(\mathsf{C}_\mathsf{b})\big)\big) = 1\Big].
\end{aligned}$$

Therefore, for all $\mathsf{D} \in \mathfrak{C}_\ell^{\text{X-}(\star)\text{ind}}$, all PPT adversaries $\mathcal{A}$, there exists a sampler $\widehat{\mathsf{D}} \in \mathfrak{C}^{\text{X-ind}}$ and a PPT adversary $\mathcal{B}$, such that

$$\mathrm{Adv}^{\text{pio-ind}(\star)}_{\mathsf{piO}_\ell^\star, \mathsf{D}, \mathcal{A}}(\lambda) = \mathrm{Adv}^{\text{pio-ind}}_{\mathsf{piO}, \widehat{\mathsf{D}}, \mathcal{B}}(\lambda + \ell(\lambda))$$

which is negligible by the security of piO for X-ind samplers and since $\widehat{\mathsf{D}} \in \mathfrak{C}^{\text{X-ind}}$.

SUPPORT RESPECTING. Let $\lambda \in \mathbb{N}$, $\mathsf{C} \in \mathcal{C}_\lambda$ and $\Lambda \in \mathit{supp}(\mathsf{piO}_\ell^\star(1^\lambda, \mathsf{C}))$. By construction of $\mathsf{piO}_\ell^\star$ and piO and perfect correctness of iO, for every input $(\mathsf{x}, \mathit{aux}) \in \{0,1\}^{n'(\lambda)} \times \{0,1\}^{\ell(\lambda)}$, we have that

$$\begin{aligned}
\Lambda(\mathsf{x}, \mathit{aux}) &= \mathcal{E}_\ell(\mathsf{C})\big(\mathsf{x}, \mathit{aux}; \mathsf{F}(\mathsf{K}, (\mathsf{x}, \mathit{aux}))\big) \\
&= \mathsf{C}\big(\mathsf{x}; \mathsf{F}(\mathsf{K}, (\mathsf{x}, \mathit{aux}))\big) \in \mathit{supp}(\mathsf{C}(\mathsf{x})).
\end{aligned}$$

STATISTICAL CORRECTNESS WITH ERROR $2^{-e(\lambda)}$. Let $e(\lambda)$ be an efficiently computable function. Let $\lambda \in \mathbb{N}$ and $\mathsf{C} \in \mathcal{C}_\lambda$ be a circuit expecting inputs $\mathsf{x} \in \{0,1\}^{n'(\lambda)}$ and randomness $\mathsf{r} \in \{0,1\}^{m(\lambda)}$, and let $\widehat{\mathsf{C}} \coloneqq \mathcal{E}_\ell(\mathsf{C})$ be its corresponding ℓ-expanded circuit. Let $\mathsf{Z} \coloneqq (\mathsf{Z}_1, \mathsf{Z}_2)$ be a joint distribution over $\{0,1\}^{n'(\lambda)} \times \{0,1\}^{\ell(\lambda)}$ with average min-entropy $\ell(\lambda) \geqslant \widetilde{\mathrm{H}}_\infty(\mathsf{Z}_2 \mid \mathsf{Z}_1) > m(\lambda) + 2e(\lambda) + 2$. Then, by Theorem 10.1, the statistical distance

$$\begin{aligned}
&\Delta\Big(\big\{\mathsf{K} \leftarrow \mathsf{KGen}(1^\lambda) \colon \big(\mathsf{K}, \mathsf{Z}_1, \mathsf{F}(\mathsf{K}, \mathsf{Z})\big)\big\}, \\
&\qquad \big\{\mathsf{K} \leftarrow \mathsf{KGen}(1^\lambda) \colon \big(\mathsf{K}, \mathsf{Z}_1, \mathsf{U}_{m(\lambda)}\big)\big\}\Big)
\end{aligned}$$

is at most $2^{-e(\lambda)}$. Since, for all distributions A, B and for every randomized function f (see, e. g., [MG02]), the statistical distance between $\mathsf{f}(\mathsf{A})$ and $\mathsf{f}(\mathsf{B})$ is upper bounded by the statistical distance between A and B, we have that the statistical distance

$$\begin{aligned}
&\Delta\Big(\big\{\mathsf{K} \leftarrow \mathsf{KGen}(1^\lambda) \colon (\mathsf{K}, \mathsf{C}(\mathsf{Z}_1; \mathsf{F}(\mathsf{K}, \mathsf{Z})))\big\}, \\
&\qquad \big\{\mathsf{K} \leftarrow \mathsf{KGen}(1^\lambda) \colon (\mathsf{K}, \mathsf{C}(\mathsf{Z}_1; \mathsf{U}_{m(\lambda)}))\big\}\Big)
\end{aligned}$$

and the statistical distance

$$\begin{aligned}
&\Delta\Big(\big\{\Lambda \leftarrow \mathsf{piO}_\ell^\star(\mathsf{C}) \colon (\Lambda, \overbrace{\mathsf{C}(\mathsf{Z}_1; \mathsf{F}(\mathsf{K}, \mathsf{Z}))}^{=\Lambda(\mathsf{Z}_1, \mathsf{Z}_2)})\big\}, \\
&\qquad \big\{\Lambda \leftarrow \mathsf{piO}_\ell^\star(\mathsf{C}) \colon (\Lambda, \mathsf{C}(\mathsf{Z}_1; \mathsf{U}_{m(\lambda)}))\big\}\Big)
\end{aligned}$$

are both upper bounded by $2^{-e(\lambda)}$. □

REMARK 10.1. Our construction also achieves security with respect to X-ind samplers. This, however, comes at the cost of a non-black-box proof reproducing the hybrid argument of [CLTV15].

11 HOW TO SIMULATE EXTRACTION

In order to instantiate the algebraic group model, we first require an abstract definitional framework of what it means to be a group in a cryptographic sense which allows to simulate crucial properties of the AGM. The notion of a *group scheme* or *encoding scheme* [GGH13] abstracts the properties of a cryptographic group. In a nutshell, a group scheme provides an interface of algorithms abstracting the handling of a cryptographic group. As we want to prove hardness of certain problems based on hardness assumptions in an already existing *base group*, we incorporate this existing group into our group scheme. Each element of our group scheme corresponds to exactly one element from a base group in a structure-preserving way. In other words, every element group of our group schemes "wraps" an element of the base group. We call the resulting notion a *group wrapper*.

For the purpose of simulating extraction properties, we introduce the notion of an *algebraic wrapper*. An algebraic wrapper is a group wrapper which provides the additional functionality that group element encodings allow for efficient extraction of representations which – similar to the algebraic group model – can be used in a security reduction. A similar approach has been taken by [KP19]. [KP19] define their group scheme as a linear subspace of $\mathbb{G} \times \mathbb{G}$ for an existing group $\mathbb{G}$ in such a way that the Generalized Knowledge of Exponent Assumption (GKEA) [WS07] can be used to extract a representation (membership can for instance be tested via a symmetric pairing). However, [KP19] relies on GKEA in the base group which more or less directly yields an equivalence between algebraic groups and GKEA. Moreover, the existence of algebraic groups implies the existence of extractable one-way functions with unbounded auxiliary input as per Definition 2.21 (since the AGM allows an additional unstructured input from $\{0,1\}^*$). This notion of extractable one-way functions conflicts with the existence of indistinguishability obfuscation [BCPR16]. More precisely, consider the additional unstructured input from $\{0,1\}^*$ encodes an obfuscated circuit which on input of a group element $[a]_G$ computes $[a]_{\mathbb{G}}^{F_K([a]_G)}$ using some hard-coded PRF key K. An adversary who simply evaluates this circuit on the group generator and outputs the result is clearly unable to explain how this output was derived algebraically. Due to this contradiction and the difficulty to assess the plausibility of knowledge-type assumptions, we strive for a weaker model which can purely be based on falsifiable assumptions.

EXTRACTION TRAPDOORS. In [KP19], extraction is possible as long as the code and the randomness which were used to produce a group element are known. Since we seek to avoid non-falsifiable knowledge-type assumptions, we need to find a different mechanism of what enables extraction. We observe that in order to reproduce proof strategies from the algebraic group model, extraction is only necessary during security reductions. Since the reduction to some assumption in the base group is in control of the parameter generation

of the wrapper, the reduction may use corresponding trapdoor information which we define to enable extraction. We call this notion *private extractability*.

In the following we provide formal definitions of group schemes and our notion of algebraic wrappers.

11.1 GROUP SCHEMES AND WRAPPERS

A group scheme or encoding scheme [GGH13] abstracts the properties of mathematical groups used in cryptography. Group schemes have recently been studied in [AFH+16; AH18; FHHL18; KP19; AMP20]. In contrast to traditional groups, group elements are not bound to be represented by some unique bitstring – henceforth referred to as *encoding*. This allows to encode auxiliary information inside group elements.

Formally, a group scheme $\mathbb{H}$ consists of the algorithms $(\mathsf{GGen}_{\mathbb{H}}, \mathsf{Sam}_{\mathbb{H}}, \mathsf{Val}_{\mathbb{H}}, \mathsf{Add}_{\mathbb{H}}, \mathsf{Eq}_{\mathbb{H}}, \mathsf{GetID}_{\mathbb{H}})$.

- $\mathsf{GGen}_{\mathbb{H}}$ is a group generation algorithm which, given 1^λ, samples group parameters $pp_{\mathbb{H}}$.
- $\mathsf{Sam}_{\mathbb{H}}$ is a sampling algorithm which, on input of the group parameters $pp_{\mathbb{H}}$ and an exponent a, produces an encoding corresponding to the exponent a.[46]
- $\mathsf{Val}_{\mathbb{H}}$ is a validation algorithm which, given the group parameters $pp_{\mathbb{H}}$ and a bitstring, decides whether the given bitstring is a valid encoding.
- $\mathsf{Add}_{\mathbb{H}}$ implements the group operation, i. e., $\mathsf{Add}_{\mathbb{H}}$ expects the group parameters $pp_{\mathbb{H}}$ and two encodings as input, and produces an encoding of the group element which results from applying the group operation on its input encodings.
- $\mathsf{Eq}_{\mathbb{H}}$ is an equality testing algorithm. Since group elements do not necessarily possess unique encodings, $\mathsf{Eq}_{\mathbb{H}}$ enables to test whether two given encodings correspond to the same group element (with respect to the given group parameters). Note that $\mathsf{Eq}_{\mathbb{H}}(pp_{\mathbb{H}}, \cdot)$ defines an equivalence relation on the set of valid bitstrings.
- $\mathsf{GetID}_{\mathbb{H}}$ is a "get-identifier" algorithm which, given the group parameters $pp_{\mathbb{H}}$ and an encoding of a group element, produces a bitstring which is unique for all encodings of the same group element.[47] This, again, compensates for the non-uniqueness of encodings.

[46] *For instance, $\mathsf{Sam}_{\mathbb{H}}$ can be implemented to simply raise a fixed group generator to the power of a.*

[47] *Previous work refers to this algorithm as extraction algorithm. However, to avoid overloading the word "extraction", we rename this algorithm in this part.*

Note that $\mathsf{Eq}_{\mathbb{H}}(pp_{\mathbb{H}}, a, b)$ can be implemented using $\mathsf{GetID}_{\mathbb{H}}$ by simply comparing $\mathsf{GetID}_{\mathbb{H}}(pp_{\mathbb{H}}, a)$ and $\mathsf{GetID}_{\mathbb{H}}(pp_{\mathbb{H}}, b)$ as bitstrings. The "get-identifier" algorithm allows to convert group elements into unique bitstrings which can, for instance, be used as keys. In some settings, group schemes are additionally equipped with a multilinear map [AFH+16; FHHL18; AMP20]. Hence, a group schemes supports all features cryptography demands from ordinary cyclic groups.

For a group scheme it is required that the quotient set

$$\left\{ a \in \{0,1\}^* \mid \mathsf{Val}_{\mathbb{H}}(pp_{\mathbb{H}}, a) = 1 \right\} / \mathsf{Eq}_{\mathbb{H}}(pp_{\mathbb{H}}, \cdot)$$

equipped with the operation defined via $\mathsf{Add}_{\mathbb{H}}(pp_{\mathbb{H}}, \cdot, \cdot)$ defines a mathematical group (with overwhelming probability over the choice of $pp_{\mathbb{H}} \leftarrow \mathsf{GGen}_{\mathbb{H}}(1^{\lambda})$). We say that a bitstring a is (an encoding of) a group element (relative to $pp_{\mathbb{H}}$), written as $a \in \mathbb{H}$, if and only if $\mathsf{Val}_{\mathbb{H}}(pp_{\mathbb{H}}, a) = 1$.

A group scheme requires that encodings corresponding to the same group element are computationally indistinguishable as formalized by the "Switching Lemma(s)" in [AFH+16; AH18; FHHL18].

We henceforth use the notation $\widehat{h}$ to mark encodings of a group element.

11.2 AN ALGEBRAIC WRAPPER

Given a cyclic group, an algebraic wrapper is a group wrapper which equips a given group $\mathbb{G}$ with a notion of extractability while preserving its group structure and computational hardness guarantees. In particular, we achieve a property which we refer to as "private extractability" with respect to a given set of group elements in the base group. More precisely, our group generation algorithm expects group parameters $pp_{\mathbb{G}}$ of the base group together with a set of group elements $[\mathbf{b}]_{\mathbb{G}} \in \mathbb{G}^n$ from that base group – henceforth referred to as *basis* – and produces group parameters $pp_{\mathbb{H}}$ of the wrapper group together with a corresponding trapdoor $\tau_{\mathbb{H}}$. We exploit the non-uniqueness of encodings to let all encodings carry some representation relative to this basis $[\mathbf{b}]_{\mathbb{G}}$. The trapdoor $\tau_{\mathbb{H}}$ enables to extract this representation from every encoding. A "representation" of a group element relative to the basis $[\mathbf{b}]_{\mathbb{G}}$ is a decomposition of the group element as linear combination of the group elements in $[\mathbf{b}]_{\mathbb{G}}$. Hence, our notion of extractability is strictly weaker than the notion provided by the algebraic group model (AGM) [FKL18], but also circumvents the implausibility results entailed by the AGM. Nonetheless, this notion suffices to simulate proof strategies developed for the algebraic group model in the very absence of knowledge-type assumptions.

More precisely, encodings which wrap some base group element $h \in \mathbb{G}$ carry (computationally hidden) representation vectors $\mathbf{z} \in \mathbb{Z}_p^n$, such that $\mathbf{z}$ contains the coefficients of a linear combination of h relative to $[\mathbf{b}]_{\mathbb{G}}$, i. e., $\mathbf{z}^{\intercal} \cdot [\mathbf{b}]_{\mathbb{G}} = h$. Private extraction allows to recover this representation vector. Given the trapdoor, we require that it is possible to "privately" sample encodings which carry some specific dictated representation vector. We require that publicly sampled encodings and privately sampled encodings are computationally indistinguishable. We refer to this property as "switching". In order to preserve tightness of security reductions when implementing AGM proofs with our algebraic wrapper, we require a statistical re-randomization property. Furthermore, we require that representation vectors compose additively (in $\mathbb{Z}_p^n$) with the group operation and do not change when encodings are re-randomized.

Let

$$\mathcal{B}^n_{pp_{\mathbb{G}}} \coloneqq \left\{ \left([1]_{\mathbb{G}}, [x_2]_{\mathbb{G}}, \ldots, [x_n]_{\mathbb{G}} \right)^{\intercal} \in \mathbb{G}^n \;\middle|\; x_2, \ldots, x_n \in \mathbb{Z}_p^{\times} \right\}$$

be the set of what we call "legitimate basis vectors". Note that we require the first group element to be the generator of the base group $[1]_{\mathbb{G}}$ to allow for public sampling (using $\mathsf{Sam}_{\mathbb{H}}$). This is necessary because $\mathsf{Sam}_{\mathbb{H}}$ must be able to encode a matching representation vector for every produced group element encoding.

DEFINITION 11.1 (Algebraic wrapper for $\mathbb{G}$). An *algebraic wrapper* $\mathbb{H}$ *for* $\mathbb{G}$ is a tuple of PPT algorithms $(\mathsf{GGen}_{\mathbb{H}}, \mathsf{Sam}_{\mathbb{H}}, \mathsf{Val}_{\mathbb{H}}, \mathsf{Add}_{\mathbb{H}}, \mathsf{Eq}_{\mathbb{H}}, \mathsf{GetID}_{\mathbb{H}}, \mathsf{Rerand}_{\mathbb{H}}, \mathsf{PrivSam}_{\mathbb{H}}, \mathsf{PrivExt}_{\mathbb{H}}, \mathsf{Unwrap}_{\mathbb{H}})$ such that the tuple $(\mathsf{GGen}_{\mathbb{H}}, \mathsf{Sam}_{\mathbb{H}}, \mathsf{Val}_{\mathbb{H}}, \mathsf{Add}_{\mathbb{H}}, \mathsf{Eq}_{\mathbb{H}}, \mathsf{GetID}_{\mathbb{H}})$ constitutes a group scheme and the following properties are satisfied.

$\mathbb{G}$-WRAPPING. The algorithm $\mathsf{Unwrap}_{\mathbb{H}}(pp_{\mathbb{H}}, \cdot)$ is deterministic and for all $pp_{\mathbb{G}} \in \mathit{supp}(\mathsf{GGen}_{\mathbb{G}}(1^\lambda))$, all legitimate basis vectors $[\mathbf{b}]_{\mathbb{G}} \in \mathcal{B}^n_{pp_{\mathbb{G}}}$, all $(pp_{\mathbb{H}}, \tau_{\mathbb{H}}) \in \mathit{supp}(\mathsf{GGen}_{\mathbb{H}}(pp_{\mathbb{G}}, [\mathbf{b}]_{\mathbb{G}}))$,

$$\mathsf{Unwrap}_{\mathbb{H}}(pp_{\mathbb{H}}, \cdot) \colon \mathbb{H} \to \mathbb{G}$$

defines a *group isomorphism*.

EXTRACTABILITY. The algorithm $\mathsf{PrivExt}_{\mathbb{H}}$ is deterministic. Furthermore, for all $pp_{\mathbb{G}} \in \mathit{supp}(\mathsf{GGen}_{\mathbb{G}}(1^\lambda))$, all legitimate basis vectors $[\mathbf{b}]_{\mathbb{G}} \in \mathcal{B}^n_{pp_{\mathbb{G}}}$, all $(pp_{\mathbb{H}}, \tau_{\mathbb{H}}) \in \mathit{supp}(\mathsf{GGen}_{\mathbb{H}}(pp_{\mathbb{G}}, [\mathbf{b}]_{\mathbb{G}}))$, all $\widehat{h} \in \mathbb{H}$, we require that $\mathsf{PrivExt}_{\mathbb{H}}$ always extracts a representation of $[x]_{\mathbb{G}} \coloneqq \mathsf{Unwrap}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{h})$ with respect to $[\mathbf{b}]_{\mathbb{G}}$, i. e., for $\mathbf{z} \coloneqq \mathsf{PrivExt}_{\mathbb{H}}(\tau_{\mathbb{H}}, \widehat{h})$,

$$[\mathbf{z}^\intercal \cdot \mathbf{b}]_{\mathbb{G}} = \mathsf{Unwrap}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{h}).$$

CORRECTNESS OF EXTRACTION. For all $pp_{\mathbb{G}} \in \mathit{supp}(\mathsf{GGen}_{\mathbb{G}}(1^\lambda))$, all $[\mathbf{b}]_{\mathbb{G}} \in \mathcal{B}^n_{pp_{\mathbb{G}}}$, all $(pp_{\mathbb{H}}, \tau_{\mathbb{H}}) \in \mathit{supp}(\mathsf{GGen}_{\mathbb{H}}(pp_{\mathbb{G}}, [\mathbf{b}]_{\mathbb{G}}))$, all $\widehat{h_0}, \widehat{h_1} \in \mathbb{H}$, we require that private extraction respects the group operation in the sense that for all $\widehat{h_2} \in \mathit{supp}(\mathsf{Add}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{h_0}, \widehat{h_1}))$, $\mathbf{z}^{(i)} \coloneqq \mathsf{PrivExt}_{\mathbb{H}}(\tau_{\mathbb{H}}, \widehat{h_i})$ for $i \in \{0, 1, 2\}$, satisfy

$$\mathbf{z}^{(2)} = \mathbf{z}^{(0)} + \mathbf{z}^{(1)}.$$

Additionally, for all $pp_{\mathbb{G}} \in \mathit{supp}(\mathsf{GGen}_{\mathbb{G}}(1^\lambda))$, all $[\mathbf{b}]_{\mathbb{G}} \in \mathcal{B}^n_{pp_{\mathbb{G}}}$, all $(pp_{\mathbb{H}}, \tau_{\mathbb{H}}) \in \mathit{supp}(\mathsf{GGen}_{\mathbb{H}}(pp_{\mathbb{G}}, [\mathbf{b}]_{\mathbb{G}}))$, all $\widehat{h} \in \mathbb{H}$, we require that re-randomization does not interfere with private extraction in the sense that for all $\widehat{h'} \in \mathit{supp}(\mathsf{Rerand}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{h}))$,

$$\mathsf{PrivExt}_{\mathbb{H}}(\tau_{\mathbb{H}}, \widehat{h}) = \mathsf{PrivExt}_{\mathbb{H}}(\tau_{\mathbb{H}}, \widehat{h'}).$$

CORRECTNESS OF SAMPLING. For all $pp_{\mathbb{G}} \in \mathit{supp}(\mathsf{GGen}_{\mathbb{G}}(1^\lambda))$, all $[\mathbf{b}]_{\mathbb{G}} \in \mathcal{B}^n_{pp_{\mathbb{G}}}$, all $(pp_{\mathbb{H}}, \tau_{\mathbb{H}}) \in \mathit{supp}(\mathsf{GGen}_{\mathbb{H}}(pp_{\mathbb{G}}, [\mathbf{b}]_{\mathbb{G}}))$, we require that

- for all $\boldsymbol{\nu} \in \mathbb{Z}_p^n$, $\Pr\left[\mathsf{PrivExt}_{\mathbb{H}}(\tau_{\mathbb{H}}, \mathsf{PrivSam}_{\mathbb{H}}(\tau_{\mathbb{H}}, \boldsymbol{\nu})) = \boldsymbol{\nu}\right] = 1$,
- for all $x \in \mathbb{Z}_p$, $\Pr\left[\mathsf{PrivExt}_{\mathbb{H}}(\tau_{\mathbb{H}}, \mathsf{Sam}_{\mathbb{H}}(pp_{\mathbb{H}}, x \cdot \boldsymbol{e_1})) = x \cdot \boldsymbol{e_1}\right] = 1$.

k-SWITCHING. We say a PPT adversary $\mathcal{A}$ is a legitimate k-switching adversary if on input of base group parameters $pp_{\mathbb{G}}$, $\mathcal{A}$ outputs two bases $([\mathbf{b}^{(j)}]_{\mathbb{G}})_{j \in \{0,1\}}$ and two lists comprising k representation vectors $(\boldsymbol{\nu}^{(j),(i)})_{i \in [k], j \in \{0,1\}}$ (and an internal state *st*) such that $[\mathbf{b}^{(0)}]_{\mathbb{G}}$, $[\mathbf{b}^{(1)}]_{\mathbb{G}} \in \mathcal{B}^n_{pp_{\mathbb{G}}}$ and $\boldsymbol{\nu}^{(0),(i)}, \boldsymbol{\nu}^{(1),(i)} \in \mathbb{Z}_p^n$ for some $n \in \mathbb{N}$ and all $i \in [k]$ and

$$\left[(\boldsymbol{\nu}^{(0),(i)})^\intercal \cdot \mathbf{b}^{(0)}\right]_{\mathbb{G}} = \left[(\boldsymbol{\nu}^{(1),(i)})^\intercal \cdot \mathbf{b}^{(1)}\right]_{\mathbb{G}}$$

for all $i \in [k]$.

$\mathrm{Exp}^{\text{k-switching}}_{\mathbb{H},\mathcal{A},b}(\lambda)$

$pp_{\mathbb{G}} \leftarrow \mathsf{GGen}_{\mathbb{G}}(1^\lambda)$

$\left(\left([\mathbf{b}^{(j)}]_{\mathbb{G}}\right)_{j\in\{0,1\}}, \left(\boldsymbol{\nu}^{(j),(i)}\right)_{i\in[k],j\in\{0,1\}}, st\right) \leftarrow \mathcal{A}(1^\lambda, pp_{\mathbb{G}})$

$(pp_{\mathbb{H}}, \tau_{\mathbb{H}}) \leftarrow \mathsf{GGen}_{\mathbb{H}}(pp_{\mathbb{G}}, [\mathbf{b}^{(b)}]_{\mathbb{G}})$

$\widehat{h}_i^* \leftarrow \mathsf{PrivSam}_{\mathbb{H}}(\tau_{\mathbb{H}}, \boldsymbol{\nu}^{(b),(i)})$

return $\mathcal{A}(pp_{\mathbb{H}}, (\widehat{h}_i^*)_{i\in[k]}, st)$

(a) Definition of the k-switching game $\mathrm{Exp}^{\text{k-switching}}_{\mathbb{H},\mathcal{A},b}(\lambda)$.

$\mathrm{Exp}^{\text{rerand}}_{\mathbb{H},\mathcal{A},b}(\lambda)$

$pp_{\mathbb{G}} \leftarrow \mathsf{GGen}_{\mathbb{G}}(1^\lambda)$

$([\mathbf{b}]_{\mathbb{G}}, st) \leftarrow \mathcal{A}(1^\lambda, pp_{\mathbb{G}})$

$(pp_{\mathbb{H}}, \tau_{\mathbb{H}}) \leftarrow \mathsf{GGen}_{\mathbb{H}}(pp_{\mathbb{G}}, [\mathbf{b}]_{\mathbb{G}})$

$(\widehat{h_0}, \widehat{h_1}, st) \leftarrow \mathcal{A}(pp_{\mathbb{H}}, \tau_{\mathbb{H}}, st)$

$\widehat{h} \leftarrow \mathsf{Rerand}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{h_b})$

return $\mathcal{A}(\widehat{h}, st)$

(b) Definition of the re-randomization game $\mathrm{Exp}^{\text{rerand}}_{\mathbb{H},\mathcal{A},b}(\lambda)$.

Figure 11.1: The re-randomization and k-switching games.

For all legitimate k-switching PPT adversaries $\mathcal{A}$,

$$\mathrm{Adv}^{\text{k-switching}}_{\mathbb{H},\mathcal{A}}(\lambda) \coloneqq \left| \Pr\left[\mathrm{Exp}^{\text{k-switching}}_{\mathbb{H},\mathcal{A},0}(\lambda) = 1\right] - \Pr\left[\mathrm{Exp}^{\text{k-switching}}_{\mathbb{H},\mathcal{A},1}(\lambda) = 1\right] \right|$$

is negligible, where $\mathrm{Exp}^{\text{k-switching}}_{\mathbb{H},\mathcal{A},b}(\lambda)$ (for $b \in \{0,1\}$) is defined in Figure 11.1a.

STATISTICALLY RE-RANDOMIZABLE. We say an unbounded adversary $\mathcal{A}$ is a legitimate re-randomization adversary if, on input of base group parameters $pp_{\mathbb{G}}$, $\mathcal{A}$ outputs $[\mathbf{b}]_{\mathbb{G}}$ and a state st such that $[\mathbf{b}]_{\mathbb{G}} \in \mathcal{B}^n_{pp_{\mathbb{G}}}$ and, in a second phase, $\mathcal{A}$ on input of $(pp_{\mathbb{H}}, \tau_{\mathbb{H}}, st)$ outputs two valid encodings $\widehat{h_0}$, $\widehat{h_1}$ (and a state st) such that

$$\mathsf{PrivExt}_{\mathbb{H}}(\tau_{\mathbb{H}}, \widehat{h_0}) = \mathsf{PrivExt}_{\mathbb{H}}(\tau_{\mathbb{H}}, \widehat{h_1}).$$

For all unbounded legitimate re-randomization adversaries $\mathcal{A}$,

$$\mathrm{Adv}^{\text{rerand}}_{\mathbb{H},\mathcal{A}}(\lambda) \coloneqq \left| \Pr\left[\mathrm{Exp}^{\text{rerand}}_{\mathbb{H},\mathcal{A},0}(\lambda) = 1\right] - \Pr\left[\mathrm{Exp}^{\text{rerand}}_{\mathbb{H},\mathcal{A},1}(\lambda) = 1\right] \right| \leqslant \frac{1}{2^\lambda},$$

where $\mathrm{Exp}^{\text{rerand}}_{\mathbb{H},\mathcal{A},b}(\lambda)$ (for $b \in \{0,1\}$) is defined in Figure 11.1b.

For simplicity, we require that encodings are always in $\{0,1\}^{p_{enc}(\lambda)}$ for a fixed polynomial $p_{enc}(\lambda)$.

The k-switching property allows to simultaneously switch the representation vectors of multiple group element encodings. It is necessary to switch all encodings simultaneously since private sampling can only be simulated knowing the trapdoor $\tau_{\mathbb{H}}$ which is not the case in $\mathrm{Exp}^{\text{k-switching}}_{\mathbb{H},\mathcal{A},b}(\lambda)$. More precisely, suppose we want to apply 1-switching twice. After switching the first representation, simulation of the resulting hybrid game requires to use private sampling and, hence, possession of the trapdoor $\tau_{\mathbb{H}}$. Consequently, a further reduction to 1-switching is not possible, since an adversary against 1-switching does not know $\tau_{\mathbb{H}}$.

Further, we note that group element encodings should always be re-randomized for the following reason. While publicly sampled encodings and

privately sampled encodings are indistinguishable due to k-switching (provided that they encode the same group elements), Definition 11.1 requires no such indistinguishability guarantee for, e. g., a privately sampled encoding and an encoding which is computed using the group operation. However, if these two encodings carry identical representation vectors, statistical re-randomizability guarantees indistinguishability if these encodings are re-randomized.

12 CONSTRUCTION

Our construction follows the ideas from [AFH+16; AH18; FHHL18]. Let $\mathsf{GGen}_{\mathbb{G}}$ be a group generator for a cyclic group $\mathbb{G}$. Let $\mathsf{Gen}_{\mathsf{SMP}}$ be a hard subset membership problem. Let $\mathsf{FHE} = (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval}, \mathsf{Rerand})$ be a perfectly correct and perfectly re-randomizable fully homomorphic public-key encryption scheme. Let $pp_{\mathbb{G}}$ be group parameters for $\mathbb{G}$ and $[\boldsymbol{\Omega}]_{\mathbb{G}} \in \mathbb{G}^n$ for some $n \in \mathbb{N}$. Let $L \subseteq X$ together with the relation R denote a subset membership problem instance sampled from $\mathsf{Gen}_{\mathsf{SMP}}$ and let $y \leftarrow X \setminus L$. Further, let pk be a public key for FHE. For ease of notation, we define $pars := (pp_{\mathbb{G}}, (L, X, R), y, pk, [\boldsymbol{\Omega}]_{\mathbb{G}})$. Let $\mathsf{NIWI} := (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify}, \mathsf{HSetup}, \mathsf{Ext})$ be a perfectly complete, perfectly extractable and perfectly witness-indistinguishable dual-mode NIWI proof system for the language

$$\mathcal{L} := \left\{ y := \left(pars, [x]_{\mathbb{G}}, C \right) \mid \exists w \colon (y, w) \in \mathcal{R} := \mathcal{R}_1 \vee \mathcal{R}_2 \vee \mathcal{R}_3 \right\}.$$

The relations $\mathcal{R}_1, \mathcal{R}_2, \mathcal{R}_3$ are defined as follows.

$$\mathcal{R}_1 = \left\{ \left((pars, [x]_{\mathbb{G}}, C), (sk, \boldsymbol{v}) \right) \;\middle|\; \begin{array}{rrcl} & \mathsf{KGen}(1^\lambda; sk) & = & (pk, sk) \\ \wedge & \mathsf{Dec}(sk, C) & = & \boldsymbol{v} \\ \wedge & [\boldsymbol{\Omega}^\intercal \cdot \boldsymbol{v}]_{\mathbb{G}} & = & [x]_{\mathbb{G}} \end{array} \right\}$$

$$\mathcal{R}_2 = \left\{ \left((pars, [x]_{\mathbb{G}}, C), (r, \boldsymbol{v}) \right) \;\middle|\; \begin{array}{rrcl} & \mathsf{Enc}(pk, \boldsymbol{v}; r) & = & C \\ \wedge & [\boldsymbol{\Omega}^\intercal \cdot \boldsymbol{v}]_{\mathbb{G}} & = & [x]_{\mathbb{G}} \end{array} \right\}$$

$$\mathcal{R}_3 = \left\{ \left((pars, [x]_{\mathbb{G}}, C), (w_y) \right) \;\middle|\; (y, w_y) \in R \right\}$$

Encodings $\widehat{h}$ of group elements in $\mathbb{H}$ are of the form

$$\widehat{h} =: \left([x]_{\mathbb{G}}, C, \pi \right),$$

where $[x]_{\mathbb{G}}$ is the "wrapped" element from the base group $\mathbb{G}$, C is a ciphertext and π is a consistency proof π produced for $\mathcal{L}$. During "honest" use of the algebraic wrapper, encodings carry proofs produced for relation $\mathcal{R}_1$ or relation $\mathcal{R}_2$. Both relations bind a "perturbed" representation vector $\boldsymbol{v}$ to the base group element $[x]_{\mathbb{G}}$. These two relations only differ in the necessary witnesses. Relation $\mathcal{R}_1$ requires the secret key which is associated with the public key in the public parameters $pp_{\mathbb{H}}$ as witness whereas relation $\mathcal{R}_2$ requires the randomness used for producing the ciphertext C. Hence, relation $\mathcal{R}_1$ can be used in settings, where the ciphertext is not produced freshly but, for instance, via homomorphic evaluation, but the secret key can be used. This will be necessary in certain obfuscated circuits. Relation $\mathcal{R}_2$, on the other hand, requires only information which is accessible for an honest user and, hence, allows public sampling of group elements. Relation $\mathcal{R}_3$ is a trapdoor branch which enables simulation. Note that during "honest" use of the algebraic wrapper $y \notin L$ and, hence, due to perfect soundness of NIWI, there exists no proof for relation $\mathcal{R}_3$.

With $m'(\lambda)$ we denote a polynomial upper bound on the number of random bits $\mathsf{FHE.Rerand}(1^\lambda, \cdot, \cdot)$ expects and with $m''(\lambda)$ we denote a polynomial upper bound on the number of random bits $\mathsf{NIWI.Prove}(1^\lambda, \cdot, \cdot, \cdot)$ expects. Let $\ell(\lambda) := m'(\lambda) + m''(\lambda) + 2(\lambda+1) + 3$. Let $\mathcal{C}_{\mathsf{add}} := \{C_{\mathsf{add}}, C^{(1)}_{\mathsf{add}}, C^{(2)}_{\mathsf{add}}, C^{(3)}_{\mathsf{add}}\}$ and let $\mathcal{C}_{\mathsf{rerand}} := \{C_{\mathsf{rerand}}, C^{(1)}_{\mathsf{rerand}}, C^{(2)}_{\mathsf{rerand}}, C^{(3)}_{\mathsf{rerand}}\}$, see Figures 12.4, 12.5 and 12.6. Let piO be a pIO scheme for the class of samplers $\mathfrak{C}^{\text{X-ind}}$ over $\mathcal{C}_{\mathsf{add}}$ and let $\mathsf{piO}^\star_\ell$ be an ℓ-expanding pIO scheme for the class of samplers $\mathfrak{C}^{\text{X-(}\star\text{)ind}}_\ell$ over $\mathcal{C}_{\mathsf{rerand}}$. Further, let $p_{\mathsf{add}}(\lambda)$ denote a polynomial upper bound on the size of addition circuits in $\mathcal{C}_{\mathsf{add}}$ and $p_{\mathsf{rerand}}(\lambda)$ denote a polynomial upper bound on the size of re-randomization circuits in $\mathcal{C}_{\mathsf{add}}$.

Our algebraic wrapper $\mathbb{H}$ is composed of the PPT algorithms $(\mathsf{GGen}_{\mathbb{H}}, \mathsf{Sam}_{\mathbb{H}}, \mathsf{Val}_{\mathbb{H}}, \mathsf{Add}_{\mathbb{H}}, \mathsf{Eq}_{\mathbb{H}}, \mathsf{Rerand}_{\mathbb{H}}, \mathsf{PrivExt}_{\mathbb{H}}, \mathsf{PrivSam}_{\mathbb{H}}, \mathsf{GetID}_{\mathbb{H}}, \mathsf{Unwrap}_{\mathbb{H}})$ defined in Figures 12.1 and 12.2. We note that the algorithm $\mathsf{Val}_{\mathbb{H}}$ which is evaluated inside C_{add} and C_{rerand} only requires a certain part of the public parameters as input. In particular, $\mathsf{Val}_{\mathbb{H}}$ does not depend on Λ_{add} and $\Lambda_{\mathsf{rerand}}$.

DIFFERENCES TO [AFH+16; AH18; FHHL18]. [AFH+16; FHHL18] introduce similar constructions of a group scheme featuring a multilinear map and of a graded encoding scheme, respectively. More precisely, [AFH+16; FHHL18] equip a base group with encodings carrying auxiliary information which can be used (in an obfuscated circuit) to "multiply in the exponent". We observe that these constructions already *wrap* a given base group in the sense that "unwrapping" encodings yields a group isomorphism to the base group.

Our construction builds upon these group schemes. In order to enable extractability with respect to a dynamically chosen basis[48], our group parameters must be generated depending on that basis.

[48] *With basis we mean a set of group elements in the base group.*

This modification, however, comes at the cost of the multilinear map functionality. This is because any implementation of a multilinear map requires knowledge of discrete logarithms of each group element encoding to a fixed generator. This is undesirable for our purposes, since we want to be able to use sets of group elements as basis which we do not know discrete logarithms of (for instance group elements provided by a reduction). Thus, we have to give up the multiplication functionality.

Furthermore, looking ahead, we crucially require that the basis can be altered via computational game hops during proofs. We solve this problem by linearly perturbing the given basis $[\mathbf{b}]_{\mathbb{G}}$ (except for its first entry to enable meaningful public sampling). We refer to this perturbed basis as $[\mathbf{\Omega}]_{\mathbb{G}}$. Our group element encodings are defined to carry representation vectors with respect to $[\mathbf{\Omega}]_{\mathbb{G}}$. By construction of C_{add}, these representation vectors are treated homomorphically by the group operation.

To preserve tightness of security reductions, we additionally introduce a statistical re-randomization mechanism.

As opposed to [AFH+16; FHHL18], [AH18] uses a quite different approach. In [AH18], the group scheme is constructed from scratch, meaning there is no necessity for an underlying group. The consequences are twofold. On one hand, very strong decisional assumptions can be proven to hold in the resulting group scheme. On the other hand, however, the group scheme from [AH18] lacks a $\mathsf{GetID}_{\mathbb{H}}$ algorithm limiting its applicability.

REMARK 12.1. We note that for our construction, we implicitly build a controlled malleable NIZK as per [CKLM12].

12.1 MAIN THEOREM AND SECURITY ANALYSIS

In the following, we formally prove that $\mathbb{H}$ as defined in Figures 12.1 and 12.2 is an algebraic wrapper.

THEOREM 12.1. *Let (i)* $\mathsf{GGen}_{\mathbb{G}}$ *be a group parameter generator for a cyclic group* $\mathbb{G}$*, (ii)* $\mathsf{Gen}_{\mathsf{SMP}}$ *be a hard subset membership problem, (iii)* $\mathsf{FHE} = (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval}, \mathsf{Rerand})$ *be a perfectly correct and perfectly re-randomizable fully homomorphic public-key encryption scheme, (iv)* $\mathsf{NIWI} := (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify}, \mathsf{HSetup}, \mathsf{Ext})$ *be a perfectly complete, perfectly sound, perfectly extractable and perfectly witness-indistinguishable dual-mode NIWI proof system for the language* $\mathcal{L}$*, (v)* piO *be a pIO scheme for the class of samplers* $\mathfrak{C}^{\text{X-ind}}$ *and (vi)* $\mathsf{piO}^{\star}_{\ell}$ *be an* ℓ*-expanding pIO scheme for the class of samplers* $\mathfrak{C}^{\text{X-}(\star)\text{ind}}_{\ell}$*. Then,* $\mathbb{H}$ *defined in Figures 12.1 and 12.2 is an algebraic wrapper.*

Proof. Since the algorithms defined in Figure 12.1 equip the base group $\mathbb{G}$ with non-unique encodings but respect its group structure, the tuple $(\mathsf{GGen}_{\mathbb{H}}, \mathsf{Sam}_{\mathbb{H}}, \mathsf{Val}_{\mathbb{H}}, \mathsf{Eq}_{\mathbb{H}}, \mathsf{Add}_{\mathbb{H}}, \mathsf{GetID}_{\mathbb{H}})$ forms a group scheme such that $\mathsf{Unwrap}_{\mathbb{H}}(pp_{\mathbb{H}}, \cdot)$ defines a group isomorphism from $\mathbb{H}$ to $\mathbb{G}$. Hence, by the correctness of FHE, piO, $\mathsf{piO}^{\star}_{\ell}$ and the completeness of NIWI, $\mathbb{H}$ satisfies $\mathbb{G}$-wrapping.

LEMMA 12.1. *The group scheme* $\mathbb{H}$ *defined in Figures 12.1 and 12.2 satisfies extractability.*

Proof of Lemma 12.1. The algorithm $\mathsf{PrivExt}_{\mathbb{H}}$ is deterministic. Let $pp_{\mathbb{G}} \in \mathit{supp}(\mathsf{GGen}_{\mathbb{G}}(1^{\lambda}))$, $[\mathbf{b}]_{\mathbb{G}} \in \mathcal{B}^{n}_{pp_{\mathbb{G}}}$ and $(pp_{\mathbb{H}}, \tau_{\mathbb{H}}) \in \mathit{supp}(\mathsf{GGen}_{\mathbb{H}}(pp_{\mathbb{G}}, [\mathbf{b}]_{\mathbb{G}}))$. Since $y \notin L$ and because NIWI is perfectly sound, every valid encoding $\widehat{h} = ([x]_{\mathbb{G}}, C, \pi)_{\mathbb{H}}$ must satisfy either relation $\mathcal{R}_1$ or $\mathcal{R}_2$ with respect to *pars*. Hence, decryption of C yields a vector $\boldsymbol{\nu}$ such that $[\boldsymbol{\Omega}^{\intercal} \cdot \boldsymbol{\nu}]_{\mathbb{G}} = [x]_{\mathbb{G}}$ or C was produced as an encryption of a vector $\boldsymbol{\nu}$ such that the above is true. Due to perfect correctness of FHE, $\mathsf{FHE.Dec}(sk, C)$ recovers this $\boldsymbol{\nu}$ in both cases. Therefore, the output z produced by $\mathsf{PrivExt}_{\mathbb{H}}(\tau_{\mathbb{H}}, \widehat{h})$ satisfies

$$\underbrace{(\nu_1 \cdot \alpha_1, \ldots, \nu_n \cdot \alpha_n)}_{=:z} \cdot [\mathbf{b}]_{\mathbb{G}} = (\nu_1, \ldots, \nu_n) \cdot \underbrace{\begin{bmatrix} b_1 \cdot \alpha_1 \\ \vdots \\ b_n \cdot \alpha_n \end{bmatrix}_{\mathbb{G}}}_{=[\boldsymbol{\Omega}]_{\mathbb{G}}} = [x]_{\mathbb{G}} .$$

Since $[x]_{\mathbb{G}} = \mathsf{Unwrap}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{h})$, extractability follows. □

LEMMA 12.2. *The group scheme* $\mathbb{H}$ *defined in Figures 12.1 and 12.2 satisfies correctness of extraction.*

$\mathsf{GGen}_{\mathbb{H}}(pp_G, [\mathbf{b}]_G)$

parse $[\mathbf{b}]_G =: [(b_1, \ldots, b_n)^\intercal]_G$
$\alpha_1 := 1, \alpha_2, \ldots, \alpha_n \leftarrow \mathbb{Z}_p^\times$
$[\boldsymbol{\Omega}]_G := ([b_1]_G^{\alpha_1}, \ldots, [b_n]_G^{\alpha_n})^\intercal$
$(pk, sk) \leftarrow \mathsf{FHE.KGen}(1^\lambda)$
$\sigma \leftarrow \mathsf{NIZK.Setup}(1^\lambda)$
$(L, X, R) \leftarrow \mathsf{Gen}_{\mathsf{SMP}}, y \leftarrow X \setminus L$
$\Lambda_{\mathsf{add}} \leftarrow \mathsf{piO}(1^{p_{\mathsf{add}}(\lambda)}, C_{\mathsf{add}})$
$\Lambda_{\mathsf{rerand}} \leftarrow \mathsf{piO}^\star_\ell(1^{p_{\mathsf{rerand}}(\lambda)}, C_{\mathsf{rerand}})$
$pars := (pp_G, (L, X, R), y, pk, [\boldsymbol{\Omega}]_G)$
$pp_{\mathbb{H}} := (\sigma, pars, \Lambda_{\mathsf{add}}, \Lambda_{\mathsf{rerand}})$
$\tau_{\mathbb{H}} := (pp_{\mathbb{H}}, sk, \alpha_1, \ldots, \alpha_n, [\mathbf{b}]_G)$
return $(pp_{\mathbb{H}}, \tau_{\mathbb{H}})$

(a) The parameter generation algorithm.

$\mathsf{Sam}_{\mathbb{H}}(pp_{\mathbb{H}}, \boldsymbol{\nu} \in \mathbb{Z}_p^n)$

$C = \mathsf{FHE.Enc}(pk, \boldsymbol{\nu}; r)$
$[x]_G := [\boldsymbol{\Omega}^\intercal \cdot \boldsymbol{\nu}]_G$
$\pi = \mathsf{Prove}(\sigma, (pars, [x]_G, C), (r, \boldsymbol{\nu}))$
return $\widehat{h} := ([x]_G, C, \pi)_{\mathbb{H}}$

(b) The (public) sampling algorithm.

$\mathsf{GetID}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{h})$

if $\neg\mathsf{Val}_{\mathbb{H}}((\sigma, pars), \widehat{h})$ **then**
  return $\perp$
parse $\widehat{h} =: ([x]_G, C, \pi)_{\mathbb{H}}$
return $[x]_G$

(c) The "extraction" algorithm producing a unique identifier.

$\mathsf{Val}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{h})$

parse $\widehat{x} =: ([x]_G, C, \pi)_{\mathbb{H}}$
return $\mathsf{Verify}(\sigma, (pars, [x]_G, C), \pi)$

(d) The validation algorithm.

$\mathsf{Eq}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{h_1}, \widehat{h_2})$

if $\exists j \in [2] \colon \neg\mathsf{Val}_{\mathbb{H}}((\sigma, pars), \widehat{h_j})$ **then**
  return $\perp$
parse $\widehat{h_i} =: ([x_i]_G, C_i, \pi_i)_{\mathbb{H}}$
return $[x_1]_G = [x_2]_G$

(e) The equality testing algorithm.

$\mathsf{Add}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{h_1}, \widehat{h_2})$

return $\Lambda_{\mathsf{add}}(\widehat{h_1}, \widehat{h_2})$

(f) The algorithm for computing the group operation.

$C_{\mathsf{add}}[pars, \sigma, sk](\widehat{h_1}, \widehat{h_2}; r)$

if $\exists j \in [2] \colon \neg\mathsf{Val}_{\mathbb{H}}((\sigma, pars), \widehat{h_j})$ **then**
  return $\perp$
parse $\widehat{h_i} =: ([x_i]_G, C_i, \pi_i)_{\mathbb{H}}$
$[x_{\mathsf{out}}]_G := [x_1]_G \cdot [x_2]_G$
$C_{\mathsf{out}} \leftarrow \mathsf{FHE.Eval}(pk, P^{(+)}[\mathbb{Z}_p^n], C_1, C_2)$
  // $P^{(+)}[\mathbb{Z}_p^n]$ computes addition in $\mathbb{Z}_p^n$
$\boldsymbol{\nu}_i \leftarrow \mathsf{FHE.Dec}(sk, C_i)$
$\boldsymbol{\nu}_{\mathsf{out}} := \boldsymbol{\nu}_1 + \boldsymbol{\nu}_2$
$\pi_{\mathsf{out}} \leftarrow \mathsf{Prove}(\sigma,$
  $(pars, [x_{\mathsf{out}}]_G, C_{\mathsf{out}}), (sk, \boldsymbol{\nu}_{\mathsf{out}}))$
return $\widehat{h_{\mathsf{out}}} := ([x_{\mathsf{out}}]_G, C_{\mathsf{out}}, \pi_{\mathsf{out}})$

(g) The group operation circuit which is obfuscated and part of the group parameters. The algorithm $\mathsf{Add}_{\mathbb{H}}$ simply evaluates this obfuscated circuit.

Figure 12.1: Definition of the algorithms $\mathsf{GGen}_{\mathbb{H}}, \mathsf{Sam}_{\mathbb{H}}, \mathsf{Val}_{\mathbb{H}}, \mathsf{Eq}_{\mathbb{H}}, \mathsf{GetID}_{\mathbb{H}}, \mathsf{Add}_{\mathbb{H}}$, $\mathsf{Unwrap}_{\mathbb{H}}$ and the circuit C_{add}.

$\mathsf{PrivSam}_{\mathbb{H}}(\tau_{\mathbb{H}}, \boldsymbol{\nu} \in \mathbb{Z}_p^n)$

$\boldsymbol{\nu}^* \coloneqq (\nu_1 \cdot \alpha_1^{-1}, \ldots, \nu_n \cdot \alpha_n^{-1})^\intercal$

$[x]_G \coloneqq [\mathbf{b}^\intercal \cdot \boldsymbol{\nu}]_G = [\boldsymbol{\Omega}^\intercal \cdot \boldsymbol{\nu}^*]_G$

$C \leftarrow \mathsf{FHE.Enc}(pk, \boldsymbol{\nu}^*; r)$

$\pi \leftarrow \mathsf{Prove}(\sigma, (pars, [x]_G, C), (sk, \boldsymbol{\nu}^*))$

return $([x]_G, C, \pi)_{\mathbb{H}}$

(a) The private sampling algorithm.

$\mathsf{PrivExt}_{\mathbb{H}}(\tau_{\mathbb{H}}, \widehat{h})$

if $\neg \mathsf{Val}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{h})$ **then**

  return $\perp$

parse $\widehat{h} =: ([x]_G, C, \pi)_{\mathbb{H}}$

$(\nu_1, \ldots, \nu_n)^\intercal =: \boldsymbol{\nu} \leftarrow \mathsf{FHE.Dec}(sk, C)$

return $(\nu_1 \cdot \alpha_1, \ldots, \nu_n \cdot \alpha_n)^\intercal$

(b) The private extraction algorithm.

$\mathsf{Unwrap}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{h})$

if $\neg \mathsf{Val}_{\mathbb{H}}((\sigma, pars), \widehat{h})$ **then**

  return $\perp$

parse $\widehat{h} =: ([x]_G, C, \pi)_{\mathbb{H}}$

return $[x]_G$

(c) The unwrap algorithm which recovers the original group element from G from encodings of group elements from $\mathbb{H}$.

$\mathsf{Rerand}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{h})$

$u \leftarrow \{0,1\}^{\ell(\lambda)}$

return $\Lambda_{\mathsf{rerand}}(\widehat{h}, u)$

(d) The re-randomization algorithm.

$C_{\mathsf{rerand}}[pars, \sigma, sk](\widehat{h}; r_1, r_2)$

if $\neg \mathsf{Val}_{\mathbb{H}}((\sigma, pars), \widehat{h})$ **then**

  return $\perp$

parse $\widehat{h} =: ([x]_G, C, \pi)_{\mathbb{H}}$

$\boldsymbol{\nu} \leftarrow \mathsf{FHE.Dec}(sk, C)$

$C_{\mathsf{out}} \leftarrow \mathsf{FHE.Rerand}(pk, C; r_1)$

$\pi_{\mathsf{out}} \leftarrow \mathsf{Prove}(\sigma,$

  $(pars, [x]_G, C_{\mathsf{out}}), (sk, \boldsymbol{\nu}); r_2)$

return $\widehat{h_{\mathsf{out}}} \coloneqq ([x]_G, C_{\mathsf{out}}, \pi_{\mathsf{out}})_{\mathbb{H}}$

(e) The re-randomization circuit which is obfuscated and part of the group parameters. The algorithm $\mathsf{Rerand}_{\mathbb{H}}$ simply evaluates this obfuscated circuit.

Figure 12.2: Definition of the algorithms $\mathsf{PrivSam}_{\mathbb{H}}, \mathsf{PrivExt}_{\mathbb{H}}, \mathsf{Rerand}_{\mathbb{H}}, \mathsf{Unwrap}_{\mathbb{H}}$ and the circuit C_{rerand}.

Proof of Lemma 12.2. We first prove, that $\mathsf{Add}_{\mathbb{H}}$ respects private extraction in $\mathbb{Z}_p^n$. Let $pp_G \in supp(\mathsf{GGen}_G(1^\lambda))$, let $[\mathbf{b}]_G \in \mathcal{B}_{pp_G}^n$ and let $(pp_{\mathbb{H}}, \tau_{\mathbb{H}}) \in supp(\mathsf{GGen}_{\mathbb{H}}(pp_G, [\mathbf{b}]_G))$. Let

$$\widehat{h_0} =: ([x_0]_G, C_0, \pi_0)_{\mathbb{H}} \in \mathbb{H} \quad \text{with} \quad \boldsymbol{\nu}^{(0)} \coloneqq \mathsf{Dec}(sk, C_0)$$
$$\widehat{h_1} =: ([x_1]_G, C_1, \pi_1)_{\mathbb{H}} \in \mathbb{H} \quad \text{with} \quad \boldsymbol{\nu}^{(1)} \coloneqq \mathsf{Dec}(sk, C_1).$$

Further, let

$$([x_2]_G, C_2, \pi_2)_{\mathbb{H}} \coloneqq \widehat{h_2} \coloneqq \mathsf{Add}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{h_0}, \widehat{h_1}) \quad \text{with} \quad \boldsymbol{\nu}^{(2)} \coloneqq \mathsf{Dec}(sk, C_2).$$

Since FHE is perfectly correct and piO is support respecting, we have $\boldsymbol{\nu}^{(2)} = \boldsymbol{\nu}^{(0)} + \boldsymbol{\nu}^{(1)}$ (in $\mathbb{Z}_p^n$). Therefore, we have

$$\underbrace{\boldsymbol{\nu}^{(2)} \circ (\alpha_1, \ldots, \alpha_n)^\intercal}_{=\mathsf{PrivExt}_{\mathbb{H}}(\tau_{\mathbb{H}}, \widehat{h_2})} = \underbrace{\boldsymbol{\nu}^{(0)} \circ (\alpha_1, \ldots, \alpha_n)^\intercal}_{=\mathsf{PrivExt}_{\mathbb{H}}(\tau_{\mathbb{H}}, \widehat{h_0})} + \underbrace{\boldsymbol{\nu}^{(1)} \circ (\alpha_1, \ldots, \alpha_n)^\intercal}_{=\mathsf{PrivExt}_{\mathbb{H}}(\tau_{\mathbb{H}}, \widehat{h_1})} \in \mathbb{Z}_p^n.$$

Note that the operator "$\circ$" denotes the Hadamard product, i. e., $(\mathbf{a} \circ \mathbf{b})_i = (\mathbf{a})_i \cdot (\mathbf{b})_i$.

Furthermore, $\mathsf{Rerand}_{\mathbb{H}}$ does not interfere with private extraction. Let $pp_G \in supp(\mathsf{GGen}_G(1^\lambda))$, $[\mathbf{b}]_G \in \mathcal{B}^n_{pp_G}$ and $(pp_{\mathbb{H}}, \tau_{\mathbb{H}}) \in supp(\mathsf{GGen}_{\mathbb{H}}(pp_G, [\mathbf{b}]_G))$. Let $\widehat{h} =: ([x]_G, C, \pi)_{\mathbb{H}} \in \mathbb{H}$ with $\nu := \mathsf{Dec}(sk, C)$. Further, let

$$([x']_G, C', \pi')_{\mathbb{H}} := \widehat{h'} \in supp(\mathsf{Rerand}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{h})) \quad \text{with}$$
$$\nu' := \mathsf{Dec}(sk, C').$$

Since FHE is perfectly correct and $\mathsf{piO}^\star_\ell$ is support respecting, $\nu = \nu'$ and thus

$$\mathsf{PrivExt}_{\mathbb{H}}(\tau_{\mathbb{H}}, \widehat{h}) = \nu \circ (\alpha_1, \ldots, \alpha_n)^\intercal = \mathsf{PrivExt}_{\mathbb{H}}(\tau_{\mathbb{H}}, \widehat{h'}).$$

□

LEMMA 12.3. *The group scheme $\mathbb{H}$ defined in Figures 12.1 and 12.2 satisfies correctness of sampling.*

Proof of Lemma 12.3. Let $pp_G \in supp(\mathsf{GGen}_G(1^\lambda))$, bases $[\mathbf{b}]_G \in \mathcal{B}^n_{pp_G}$ and $(pp_{\mathbb{H}}, \tau_{\mathbb{H}}) \in supp(\mathsf{GGen}_{\mathbb{H}}(pp_G, [\mathbf{b}]_G))$.

We prove that $\mathsf{PrivExt}_{\mathbb{H}}(\tau_{\mathbb{H}}, \cdot)$ and $\mathsf{PrivSam}_{\mathbb{H}}(\tau_{\mathbb{H}}, \cdot)$ act inversely to each other. Let $\nu \in \mathbb{Z}_p^n$ and let $([x]_G, C, \pi)_{\mathbb{H}} := \widehat{h} \in supp(\mathsf{PrivSam}_{\mathbb{H}}(\tau_{\mathbb{H}}, \nu))$. By correctness of FHE, $\mathsf{FHE.Dec}(sk, C) = \nu \circ (\alpha_1^{-1}, \ldots, \alpha_n^{-1})^\intercal$. Hence, the algorithm $\mathsf{PrivExt}_{\mathbb{H}}(\tau_{\mathbb{H}}, \widehat{h})$ outputs $((\nu_1 \cdot \alpha_1^{-1}) \cdot \alpha_1, \ldots, (\nu_n \cdot \alpha_n^{-1}) \cdot \alpha_n)^\intercal = \nu$.

Next, we prove that $\mathsf{PrivExt}_{\mathbb{H}}(\tau_{\mathbb{H}}, \cdot)$ and $\mathsf{Sam}_{\mathbb{H}}(pp_{\mathbb{H}}, \cdot)$ act inversely to each other for multiples of $\boldsymbol{e_1}$. Let $x \in \mathbb{Z}_p$ and let $([x]_G, C, \pi)_{\mathbb{H}} := \widehat{h} \in supp(\mathsf{Sam}_{\mathbb{H}}(pp_{\mathbb{H}}, x \cdot \boldsymbol{e_1}))$. By correctness of FHE, $\mathsf{FHE.Dec}(sk, C) = x \cdot \boldsymbol{e_1} \circ (\alpha_1^{-1}, \ldots, \alpha_n^{-1})^\intercal = x \cdot \boldsymbol{e_1}$ since $\alpha_1 = 1$. Hence, $\mathsf{PrivExt}_{\mathbb{H}}(\tau_{\mathbb{H}}, \widehat{h})$ outputs $(x \cdot \alpha_1, 0, \ldots, 0)^\intercal = x \cdot \boldsymbol{e_1}$ since $\alpha_1 = 1$. □

LEMMA 12.4 (Removing information from $pp_{\mathbb{H}}$). *Let $\mathsf{GGen}'_{\mathbb{H}}$ denote the distribution of public parameters sampled as in $\mathsf{GGen}_{\mathbb{H}}$ with the difference that $y \leftarrow L$ is a yes-instance of the subset membership problem, σ is sampled according to HSetup and Λ_{add}, $\Lambda_{\mathsf{rerand}}$ are produced for the circuits $C^{(3)}_{\mathsf{add}}$ and $C^{(3)}_{\mathsf{rerand}}$, see Figures 12.4, 12.5 and 12.6. Then, for all legitimate PPT adversaries $\mathcal{A}$,*

$$\mathrm{Adv}^{\mathsf{swap}}_{\mathbb{H},\mathcal{A}}(\lambda) := \left| \Pr\left[\mathrm{Exp}^{\mathsf{swap}}_{\mathbb{H},\mathcal{A},0}(\lambda) = 1\right] - \Pr\left[\mathrm{Exp}^{\mathsf{swap}}_{\mathbb{H},\mathcal{A},1}(\lambda) = 1\right] \right|$$

is negligible, where $\mathrm{Exp}^{\mathsf{swap}}_{\mathbb{H},\mathcal{A},b}(\lambda)$ is defined in Figure 12.3 and where legitimate means that $\mathcal{A}$ on input of pp_G guarantees $[\mathbf{b}]_G \in \mathcal{B}^n_{pp_G}$ for some $n \in \mathbb{N}$.

Proof of Lemma 12.4. The proof strategy is closely related to the proof of the "swap lemma" from [AFH+16; AH18]. After some preparations, we replace the no-instance of the subset membership problem with a yes-instance $y \in L$ enabling relation $\mathcal{R}_3$ to be satisfied. The extraction trapdoor td_{ext} for σ used inside the obfuscated circuits can then be replaced by the unique witness w_y for $(y, w_y) \in R$. This enables to switch σ from binding to hiding mode. Due to perfect witness-indistinguishability, the obfuscated circuits can be replaced by variants which *always* use w_y to simulate proofs. We refer the reader to Table 12.1 for an overview of the hybrid games. We assume that the circuits C_{add}, $C^{(1)}_{\mathsf{add}}$, $C^{(2)}_{\mathsf{add}}$, $C^{(3)}_{\mathsf{add}}$ are padded to size $p_{\mathsf{add}}(\lambda)$ and the circuits C_{rerand},

$\mathrm{Exp}^{\mathsf{swap}}_{\mathbb{H},\mathcal{A},0}(\lambda)$

$pp_{\mathbb{G}} \leftarrow \mathsf{GGen}_{\mathbb{G}}(1^{\lambda})$
$([\mathbf{b}]_{\mathbb{G}}, st) \leftarrow \mathcal{A}(1^{\lambda}, pp_{\mathbb{G}})$
$(pp_{\mathbb{H}}, \tau_{\mathbb{H}}) \leftarrow \mathsf{GGen}_{\mathbb{H}}(pp_{\mathbb{G}}, [\mathbf{b}]_{\mathbb{G}})$
return $\mathcal{A}(pp_{\mathbb{H}}, \tau_{\mathbb{H}}, st)$

(a) In game $\mathrm{Exp}^{\mathsf{swap}}_{\mathbb{H},\mathcal{A},0}(\lambda)$, $\mathcal{A}$ has access to parameters distributed according to $\mathsf{GGen}_{\mathbb{H}}$.

$\mathrm{Exp}^{\mathsf{swap}}_{\mathbb{H},\mathcal{A},1}(\lambda)$

$pp_{\mathbb{G}} \leftarrow \mathsf{GGen}_{\mathbb{G}}(1^{\lambda})$
$([\mathbf{b}]_{\mathbb{G}}, st) \leftarrow \mathcal{A}(1^{\lambda}, pp_{\mathbb{G}})$
$(pp_{\mathbb{H}}, \tau_{\mathbb{H}}) \leftarrow \mathsf{GGen}'_{\mathbb{H}}(pp_{\mathbb{G}}, [\mathbf{b}]_{\mathbb{G}})$
return $\mathcal{A}(pp_{\mathbb{H}}, \tau_{\mathbb{H}}, st)$

(b) In game $\mathrm{Exp}^{\mathsf{swap}}_{\mathbb{H},\mathcal{A},1}(\lambda)$, $\mathcal{A}$ has access to parameters distributed according to $\mathsf{GGen}'_{\mathbb{H}}$.

Figure 12.3: Indistinguishability between $\mathsf{GGen}_{\mathbb{H}}$ and $\mathsf{GGen}'_{\mathbb{H}}$.

Table 12.1: Overview of proof steps of Lemma 12.4. Changes to previous games are highlighted in boxes.

	$y \in L$	Λ_{add}	$\Lambda_{\mathsf{rerand}}$	σ	Remark
$\mathbf{G}_0$	NO	C_{add}	C_{rerand}	Setup	
$\mathbf{G}_2$	NO	$\boxed{C^{(1)}_{\mathsf{add}}}$	$\boxed{C^{(1)}_{\mathsf{rerand}}}$	Setup	piO and $\mathsf{piO}^{\star}_{\ell}$
$\mathbf{G}_3$	$\boxed{\text{YES}}$	$C^{(1)}_{\mathsf{add}}$	$C^{(1)}_{\mathsf{rerand}}$	Setup	subset membership problem
$\mathbf{G}_5$	YES	$\boxed{C^{(2)}_{\mathsf{add}}}$	$\boxed{C^{(2)}_{\mathsf{rerand}}}$	Setup	piO and $\mathsf{piO}^{\star}_{\ell}$
$\mathbf{G}_6$	YES	$C^{(2)}_{\mathsf{add}}$	$C^{(2)}_{\mathsf{rerand}}$	$\boxed{\mathsf{HSetup}}$	CRS indistinguishability
$\mathbf{G}_8$	YES	$\boxed{C^{(3)}_{\mathsf{add}}}$	$\boxed{C^{(3)}_{\mathsf{rerand}}}$	HSetup	piO and $\mathsf{piO}^{\star}_{\ell}$

$C^{(1)}_{\mathsf{rerand}}$, $C^{(2)}_{\mathsf{rerand}}$, $C^{(3)}_{\mathsf{rerand}}$ are padded to size $p_{\mathsf{rerand}}(\lambda)$, see Figures 12.4, 12.5 and 12.6.

GAME $\mathbf{G}_0$. This game is identical to $\mathrm{Exp}^{\mathsf{swap}}_{\mathbb{H},\mathcal{A},0}(\lambda)$.

GAME $\mathbf{G}_1$. $\mathbf{G}_1$ is defined as $\mathbf{G}_0$ but Λ_{add} is produced via $\mathsf{piO}(1^{p_{\mathsf{add}}(\lambda)}, C^{(1)}_{\mathsf{add}})$, see Figure 12.4.

Indistinguishability between $\mathbf{G}_0$ and $\mathbf{G}_1$ follows from the security of piO with respect to $\mathfrak{C}^{\text{X-ind}}$. More formally, let $\mathcal{A}$ be a legitimate PPT adversary distinguishing $\mathbf{G}_0$ and $\mathbf{G}_1$.

Let D_1 be the circuit sampler which on input of $1^{p_{\mathsf{add}}(\lambda)}$ samples $pp_{\mathbb{G}} \leftarrow \mathsf{GGen}_{\mathbb{G}}(1^{\lambda})$, calls $\mathcal{A}(pp_{\mathbb{G}})$ to obtain $([\mathbf{b}]_{\mathbb{G}}, st)$, produces parameters exactly as $\mathsf{GGen}_{\mathbb{H}}(pp_{\mathbb{G}}, [\mathbf{b}]_{\mathbb{G}})$ (except for the obfuscated circuit Λ_{add}), and outputs $C_0 := C_{\mathsf{add}}$, $C_1 := C^{(1)}_{\mathsf{add}}$ and auxiliary information $z := (\sigma, pars = (pp_{\mathbb{G}}, (L, X, R), y, pk, [\mathbf{\Omega}]_{\mathbb{G}}), \Lambda_{\mathsf{rerand}}, (sk, \alpha_1, \ldots, \alpha_n, [\mathbf{b}]_{\mathbb{G}}), st)$. Since $y \notin L$ and NIWI is perfectly sound, the circuits C_{add} and $C^{(1)}_{\mathsf{add}}$ behave exactly identically on identical inputs and random tapes. Hence, for the function $X(\lambda) := 0$ and a differing domain $\mathcal{X} := \varnothing$, the circuit sampler D_1 satisfies X-differing inputs and X-indistinguishability and, hence, is an X-ind sampler.

We construct an adversary $\mathcal{B}_1$ on the security of piO. On input of $(1^{p_{\mathsf{add}}(\lambda)}, C_0, C_1, z, \Lambda)$, $\mathcal{B}_1$ defines $pp_{\mathbb{H}} := (\sigma, pars, \Lambda, \Lambda_{\mathsf{rerand}})$ and $\tau_{\mathbb{H}} := (pp_{\mathbb{H}}, sk, \alpha_1, \ldots, \alpha_n, [\mathbf{b}]_{\mathbb{G}})$, calls $\mathcal{A}(pp_{\mathbb{H}}, \tau_{\mathbb{H}}, st)$ and outputs $\mathcal{A}$'s output. If Λ is produced via $\mathsf{piO}(1^{p_{\mathsf{add}}(\lambda)}, C_{\mathsf{add}})$, D_1 together with $\mathcal{B}_1$ perfectly simulate $\mathbf{G}_0$ for $\mathcal{A}$.

Otherwise, if Λ is produced via $\mathsf{piO}(1^{p_{\mathsf{add}}(\lambda)}, C^{(1)}_{\mathsf{add}})$, D_1 together with $\mathcal{B}_1$ perfectly simulate $\mathbf{G}_1$ for $\mathcal{A}$. Hence,

$$\left|\Pr\left[out_1 = 1\right] - \Pr\left[out_0 = 1\right]\right| \leqslant \mathrm{Adv}^{\mathsf{pio\text{-}ind}}_{\mathsf{piO},D_1,\mathcal{B}_1}(p_{\mathsf{add}}(\lambda)).$$

GAME $\mathbf{G}_2$. The game $\mathbf{G}_2$ is identical to $\mathbf{G}_1$ except that $\Lambda_{\mathsf{rerand}}$ is produced via $\mathsf{piO}^\star_\ell(1^{p_{\mathsf{rerand}}(\lambda)}, C^{(1)}_{\mathsf{rerand}})$, see Figure 12.4. Indistinguishability between $\mathbf{G}_1$ and $\mathbf{G}_2$ follows from the security of $\mathsf{piO}^\star_\ell$ with respect to $\mathfrak{C}^{\mathsf{X\text{-}(\star)ind}}_\ell$. The proof is very similar to the previous game hop. The main difference is that the used circuit sampler needs to be in $\mathfrak{C}^{\mathsf{X\text{-}(\star)ind}}_\ell$.

Let $\mathcal{A}$ be a legitimate PPT adversary distinguishing $\mathbf{G}_1$ and $\mathbf{G}_2$. Let D_2 be the circuit sampler which on input of $1^{p_{\mathsf{rerand}}(\lambda)}$ samples $pp_{\mathbb{G}} \leftarrow \mathsf{GGen}_{\mathbb{G}}(1^\lambda)$, calls $\mathcal{A}(pp_{\mathbb{G}})$ to obtain $([\mathbf{b}]_{\mathbb{G}}, st)$ and produces parameters as $\mathsf{GGen}_{\mathbb{H}}(pp_{\mathbb{G}}, [\mathbf{b}]_{\mathbb{G}})$ (except for the obfuscated circuit $\Lambda_{\mathsf{rerand}}$ and such that Λ_{add} is produced for $C^{(1)}_{\mathsf{add}}$), and outputs $C_0 := C_{\mathsf{rerand}}$, $C_1 := C^{(1)}_{\mathsf{rerand}}$ and auxiliary information $z := (\sigma, pars = (pp_{\mathbb{G}}, (L, X, R), y, pk, [\mathbf{\Omega}]_{\mathbb{G}}), \Lambda_{\mathsf{add}}, (sk, \alpha_1, \ldots, \alpha_n, [\mathbf{b}]_{\mathbb{G}}), st)$. Since $y \notin L$ and NIWI is perfectly sound, the circuits C_{rerand} and $C^{(1)}_{\mathsf{rerand}}$ behave exactly identically on identical inputs and random tapes. Therefore, by Corollary 10.1, $D_2 \in \mathfrak{C}^{\mathsf{X\text{-}(\star)ind}}_\ell$. Hence, there exists a PPT adversary $\mathcal{B}_2$ such that

$$\left|\Pr\left[out_2 = 1\right] - \Pr\left[out_1 = 1\right]\right| \leqslant \mathrm{Adv}^{\mathsf{pio\text{-}ind}(\star)}_{\mathsf{piO}^\star_\ell,D_2,\mathcal{B}_2}(p_{\mathsf{rerand}}(\lambda)).$$

GAME $\mathbf{G}_3$. The game $\mathbf{G}_3$ is defined as game $\mathbf{G}_2$ except that $y \leftarrow L$ is a yes-instance of the subset membership problem. This game hop is justified by the hardness of the subset membership problem. More precisely, for every PPT distinguisher $\mathcal{A}$ between $\mathbf{G}_2$ and $\mathbf{G}_3$, there exists an adversary $\mathcal{B}_3$ such that

$$\left|\Pr\left[out_3 = 1\right] - \Pr\left[out_2 = 1\right]\right| \leqslant \mathrm{Adv}^{\mathsf{smp}}_{\mathsf{Gen}_{\mathsf{SMP}},\mathcal{B}_3}(\lambda).$$

GAME $\mathbf{G}_4$. $\mathbf{G}_4$ is defined as $\mathbf{G}_3$ but Λ_{add} is produced via $\mathsf{piO}(1^{p_{\mathsf{add}}(\lambda)}, C^{(2)}_{\mathsf{add}})$, see Figure 12.5. Due to the perfect extractability of NIWI and the fact that for every $y \in L$ there exists exactly one witness w_y for the statement $y \in L$, the two circuits $C^{(1)}_{\mathsf{add}}$ and $C^{(2)}_{\mathsf{add}}$ behave exactly identically on identical inputs and random tapes. Hence, for every PPT adversary $\mathcal{A}$, there exists a circuit sampler $D_4 \in \mathfrak{C}^{\mathsf{X\text{-}ind}}$ and a PPT adversary $\mathcal{B}_4$ such that

$$\left|\Pr\left[out_4 = 1\right] - \Pr\left[out_3 = 1\right]\right| \leqslant \mathrm{Adv}^{\mathsf{pio\text{-}ind}}_{\mathsf{piO},D_4,\mathcal{B}_4}(p_{\mathsf{add}}(\lambda)).$$

GAME $\mathbf{G}_5$. The game $\mathbf{G}_5$ is identical to $\mathbf{G}_4$ except that $\Lambda_{\mathsf{rerand}}$ is produced via $\mathsf{piO}^\star_\ell(1^{p_{\mathsf{rerand}}(\lambda)}, C^{(2)}_{\mathsf{rerand}})$, see Figure 12.5. Indistinguishability between $\mathbf{G}_4$ and $\mathbf{G}_5$ follows from the security of $\mathsf{piO}^\star_\ell$ with respect to $\mathfrak{C}^{\mathsf{X\text{-}(\star)ind}}_\ell$. Again, due to the perfect extractability of NIWI and the fact that for every $y \in L$ there exists exactly one witness w_y for the statement $y \in L$, the two circuits $C^{(1)}_{\mathsf{rerand}}$ and $C^{(2)}_{\mathsf{rerand}}$ behave exactly identically on identical inputs and random tapes. Hence, by Corollary 10.1, for every PPT adversary $\mathcal{A}$, there exists a circuit sampler $D_5 \in \mathfrak{C}^{\mathsf{X\text{-}(\star)ind}}_\ell$ and a PPT adversary $\mathcal{B}_5$ such that

$$\left|\Pr\left[out_5 = 1\right] - \Pr\left[out_4 = 1\right]\right| \leqslant \mathrm{Adv}^{\mathsf{pio\text{-}ind}(\star)}_{\mathsf{piO}^\star_\ell,D_5,\mathcal{B}_5}(p_{\mathsf{rerand}}(\lambda)).$$

$C^{(1)}_{\mathsf{add}}[pars, \sigma, sk, td_{ext}](\widehat{h_1}, \widehat{h_2})$

if $\exists j \in [2] \colon \neg \mathsf{Val}_{\mathbb{H}}((\sigma, pars), \widehat{h_j})$ **then**
  return $\perp$
parse $\widehat{h_i} =: ([x_i]_G, C_i, \pi_i)_{\mathbb{H}}$
$[x_{out}]_G \coloneqq [x_1]_G \cdot [x_2]_G$
$C_{out} \leftarrow \mathsf{FHE.Eval}(pk, P^{(+)}[\mathbb{Z}_p^n], C_1, C_2)$
$v_i \leftarrow \mathsf{Dec}(sk, C_i)$
$v_{out} \coloneqq v_1 + v_2$
if $[\Omega^\intercal \cdot v_{out}]_G = [x_{out}]_G$ **then**
  $\pi_{out} \leftarrow \mathsf{Prove}(\sigma,$
    $(pars, [x_{out}]_G, C_{out}), (sk, v_{out}))$
else
  let $j \in \{1,2\}$ s.t. $[\Omega]_G^\intercal \cdot v_j \neq [x_j]_G$
  $w_y \leftarrow \mathsf{Ext}(td_{ext}, (pars, [x_j]_G, C_j), \pi_j)$
  $\pi_{out} \leftarrow \mathsf{Prove}(\sigma,$
    $(pars, [x_{out}]_G, C_{out}), w_y)$
return $\widehat{h_{out}} \coloneqq ([x_{out}]_G, C_{out}, \pi_{out})_{\mathbb{H}}$

(a) Addition circuit $C^{(1)}_{\mathsf{add}}$.

$C^{(1)}_{\mathsf{rerand}}[pars, \sigma, sk, td_{ext}](\widehat{h})$

if $\neg \mathsf{Val}_{\mathbb{H}}((\sigma, pars), \widehat{h})$ **then**
  return $\perp$
parse $\widehat{h} =: ([x]_G, C, \pi)_{\mathbb{H}}$
$v \leftarrow \mathsf{Dec}(sk, C)$
$C_{out} \leftarrow \mathsf{FHE.Rerand}(pk, C)$
if $[\Omega^\intercal \cdot v]_G = [x]_G$ **then**
  $\pi_{out} \leftarrow \mathsf{Prove}(\sigma,$
    $(pars, [x]_G, C_{out}), (sk, v))$
else
  $w_y \leftarrow \mathsf{Ext}(td_{ext}, (pars, [x]_G, C), \pi)$
  $\pi_{out} \leftarrow \mathsf{Prove}(\sigma,$
    $(pars, [x]_G, C_{out}), w_y)$
return $\widehat{h_{out}} \coloneqq ([x]_G, C_{out}, \pi_{out})_{\mathbb{H}}$

(b) Re-randomization circuit $C^{(1)}_{\mathsf{rerand}}$.

Figure 12.4: Addition and re-randomization circuits $C^{(1)}_{\mathsf{add}}$ and $C^{(1)}_{\mathsf{rerand}}$ which are capable to deal with simulated proofs by extracting a witness for relation $\mathcal{R}_3$.

Thus, $\mathbf{G}_5$ does not make use of the extraction trapdoor td_{ext} corresponding to σ anymore.

GAME $\mathbf{G}_6$. The game $\mathbf{G}_6$ is defined as $\mathbf{G}_5$, except that σ is produced via $\mathsf{HSetup}(1^\lambda)$, i. e., in hiding mode. This game hop is justified by CRS indistinguishability of NIWI. Hence, there exists a PPT adversary $\mathcal{B}_6$ such that

$$\left|\Pr\left[out_6 = 1\right] - \Pr\left[out_5 = 1\right]\right| \leqslant \mathrm{Adv}^{\mathsf{crs\text{-}ind}}_{\mathsf{NIWI},\mathcal{B}_6}(\lambda).$$

GAME $\mathbf{G}_7$. $\mathbf{G}_7$ is defined as $\mathbf{G}_6$ but Λ_{add} is produced via $\mathsf{piO}(1^{p_{\mathsf{add}}(\lambda)}, C^{(3)}_{\mathsf{add}})$, see Figure 12.6. Let D_7 be a circuit sampler, which produces public parameters as in $\mathbf{G}_6$ and outputs the circuits $C_0 \coloneqq C^{(2)}_{\mathsf{add}}$ and $C_1 \coloneqq C^{(3)}_{\mathsf{add}}$ (and suitable auxiliary information z). The circuit sampler D_7 is an X-ind sampler for $X(\lambda) \coloneqq 2^{2p_{enc}(\lambda)} \leqslant 2^{p_{\mathsf{add}}(\lambda)}$ and $\mathcal{X} \coloneqq \{0,1\}^{2p_{enc}(\lambda)}$ since the X-differing inputs property is trivially satisfied and X-indistinguishability is satisfied since perfect witness indistinguishability of NIWI implies that for every input $x = (\widehat{a_1}, \widehat{a_2}) \in \{0,1\}^{2p_{enc}(\lambda)}$, the output distributions produced by $C^{(2)}_{\mathsf{add}}(x)$ and $C^{(3)}_{\mathsf{add}}(x)$ are identical. Hence, for every PPT adversary $\mathcal{A}$, there exists a circuit sampler $D_7 \in \mathfrak{C}^{\text{X-ind}}$ and a PPT adversary $\mathcal{B}_7$ such that

$$\left|\Pr\left[out_7 = 1\right] - \Pr\left[out_6 = 1\right]\right| \leqslant \mathrm{Adv}^{\mathsf{pio\text{-}ind}}_{\mathsf{piO},D_7,\mathcal{B}_7}(p_{\mathsf{add}}(\lambda)).$$

$\mathsf{C}^{(2)}_{\mathsf{add}}[pars, \sigma, sk, w_{\mathcal{Y}}](\widehat{h_1}, \widehat{h_2})$

if $\exists j \in [2]: \neg \mathsf{Val}_{\mathbb{H}}((\sigma, pars), \widehat{h_j})$ **then**
  return $\perp$
parse $\widehat{h_i} =: ([x_i]_G, C_i, \pi_i)_{\mathbb{H}}$
$[x_{out}]_G := [x_1]_G \cdot [x_2]_G$
$C_{out} \leftarrow \mathsf{FHE.Eval}(pk, P^{(+)}[\mathbb{Z}_p^n], C_1, C_2)$
$\nu_i \leftarrow \mathsf{Dec}(sk, C_i)$
$\nu_{out} := \nu_1 + \nu_2$
if $[\Omega^\intercal \cdot \nu_{out}]_G = [x_{out}]_G$ **then**
  $\pi_{out} \leftarrow \mathsf{Prove}(\sigma,$
    $(pars, [x_{out}]_G, C_{out}), (sk, \nu_{out}))$
else
  $\pi_{out} \leftarrow \mathsf{Prove}(\sigma,$
    $(pars, [x_{out}]_G, C_{out}), w_{\mathcal{Y}})$
return $\widehat{h_{out}} := ([x_{out}]_G, C_{out}, \pi_{out})_{\mathbb{H}}$

(a) Addition circuit $\mathsf{C}^{(2)}_{\mathsf{add}}$.

$\mathsf{C}^{(2)}_{\mathsf{rerand}}[pars, \sigma, sk, w_{\mathcal{Y}}](\widehat{h})$

if $\neg \mathsf{Val}_{\mathbb{H}}((\sigma, pars), \widehat{h})$ **then**
  return $\perp$
parse $\widehat{h} =: ([x]_G, C, \pi)_{\mathbb{H}}$
$\nu \leftarrow \mathsf{Dec}(sk, C)$
$C_{out} \leftarrow \mathsf{FHE.Rerand}(pk, C)$
if $[\Omega^\intercal \cdot \nu]_G = [x]_G$ **then**
  $\pi_{out} \leftarrow \mathsf{Prove}(\sigma,$
    $(pars, [x]_G, C_{out}), (sk, \nu))$
else
  $\pi_{out} \leftarrow \mathsf{Prove}(\sigma,$
    $(pars, [x]_G, C_{out}), w_{\mathcal{Y}})$
return $\widehat{h_{out}} := ([x]_G, C_{out}, \pi_{out})_{\mathbb{H}}$

(b) Re-randomization circuit $\mathsf{C}^{(2)}_{\mathsf{rerand}}$.

Figure 12.5: Addition and re-randomization circuits $\mathsf{C}^{(2)}_{\mathsf{add}}$ and $\mathsf{C}^{(2)}_{\mathsf{rerand}}$ which use a hard-coded witness $w_{\mathcal{Y}}$ for relation $\mathcal{R}_3$.

GAME $\mathbf{G}_8$. The game $\mathbf{G}_8$ is identical to $\mathbf{G}_7$ except that $\Lambda_{\mathsf{rerand}}$ is produced via $\mathsf{piO}^\star_\ell(1^{p_{\mathsf{rerand}}(\lambda)}, \mathsf{C}^{(3)}_{\mathsf{rerand}})$, see Figure 12.6. Indistinguishability between $\mathbf{G}_7$ and $\mathbf{G}_8$ follows from the security of $\mathsf{piO}^\star_\ell$ with respect to $\mathfrak{C}^{\mathsf{X}\text{-}(\star)\mathsf{ind}}_\ell$. Again, due to the perfect witness indistinguishability of NIWI we have that for every input $x = \widehat{a} \in \{0,1\}^{p_{enc}(\lambda)}$, the output distributions produced by $\mathsf{C}^{(2)}_{\mathsf{rerand}}(x)$ and $\mathsf{C}^{(3)}_{\mathsf{rerand}}(x)$ are identical.

Let D_8 be the circuit sampler which on input of $1^{p_{\mathsf{rerand}}(\lambda)}$ samples $pp_G \leftarrow \mathsf{GGen}_G(1^\lambda)$, calls $\mathcal{A}(pp_G)$ to obtain $([\mathbf{b}]_G, st)$ and produces parameters as in $\mathbf{G}_7$ (except for the obfuscated circuit $\Lambda_{\mathsf{rerand}}$) and outputs $C_0 := \mathsf{C}^{(2)}_{\mathsf{rerand}}$, $C_1 := \mathsf{C}^{(3)}_{\mathsf{rerand}}$ and auxiliary information $z := (\sigma, pars = (pp_G, (L, X, R), y, pk, [\mathbf{\Omega}]_G), \Lambda_{\mathsf{add}}, (sk, \alpha_1, \ldots, \alpha_n, [\mathbf{b}]_G), st)$. By Lemma 10.1, $\mathsf{D}_8 \in \mathfrak{C}^{\mathsf{X}\text{-}(\star)\mathsf{ind}}_\ell$ and, thus, for every PPT adversary $\mathcal{A}$, there exists a circuit sampler $\mathsf{D}_8 \in \mathfrak{C}^{\mathsf{X}\text{-}(\star)\mathsf{ind}}_\ell$ and a PPT adversary $\mathcal{B}_8$ such that

$$\left|\Pr\left[out_8 = 1\right] - \Pr\left[out_7 = 1\right]\right| \leqslant \mathsf{Adv}^{\mathsf{pio\text{-}ind}(\star)}_{\mathsf{piO}^\star_\ell, \mathsf{D}_8, \mathcal{B}_8}(p_{\mathsf{rerand}}(\lambda))$$

is negligible.

Note that $\mathbf{G}_8$ is defined as $\mathsf{Exp}^{\mathsf{swap}}_{\mathbb{H}, \mathcal{A}, 1}(\lambda)$. Therefore,

$$\mathsf{Adv}^{\mathsf{swap}}_{\mathbb{H}, \mathcal{A}}(\lambda) \leqslant \left|\Pr\left[out_0 = 1\right] - \Pr\left[out_8 = 1\right]\right|$$

is negligible in λ. □

LEMMA 12.5. *The group scheme $\mathbb{H}$ defined in Figures 12.1 and 12.2 satisfies* k*-switching.*

$C^{(3)}_{\mathsf{add}}[pars, \sigma, sk, w_y](\widehat{h_1}, \widehat{h_2})$

if $\exists j \in [2] : \neg \mathsf{Val}_{\mathbb{H}}((\sigma, pars), \widehat{h_j})$ **then**
  return $\perp$
parse $\widehat{h_i} =: ([x_i]_G, C_i, \pi_i)_{\mathbb{H}}$
$[x_{out}]_G := [x_1]_G \cdot [x_2]_G$
$C_{out} \leftarrow \mathsf{FHE.Eval}(pk, P^{(+)}[\mathbb{Z}_p^n], C_1, C_2)$
$\pi_{out} \leftarrow \mathsf{Prove}(\sigma,$
  $(pars, [x_{out}]_G, C_{out}), w_y)$
return $\widehat{h_{out}} := ([x_{out}]_G, C_{out}, \pi_{out})_{\mathbb{H}}$

(a) Addition circuit $C^{(3)}_{\mathsf{add}}$.

$C^{(3)}_{\mathsf{rerand}}[pars, \sigma, sk, w_y](\widehat{h})$

if $\neg \mathsf{Val}_{\mathbb{H}}((\sigma, pars), \widehat{h})$ **then**
  return $\perp$
parse $\widehat{h} =: ([x]_G, C, \pi)_{\mathbb{H}}$
$C_{out} \leftarrow \mathsf{FHE.Rerand}(pk, C)$
$\pi_{out} \leftarrow \mathsf{Prove}(\sigma,$
  $(pars, [x]_G, C_{out}), w_y)$
return $\widehat{h_{out}} := ([x]_G, C_{out}, \pi_{out})_{\mathbb{H}}$

(b) Re-randomization circuit $C^{(3)}_{\mathsf{rerand}}$.

Figure 12.6: Addition and re-randomization circuits $C^{(3)}_{\mathsf{add}}$ and $C^{(3)}_{\mathsf{rerand}}$ which always produce consistency proofs for relation $\mathcal{R}_3$.

Proof of Lemma 12.5. We recall that an adversary $\mathcal{A}$ is a legitimate k-switching adversary if $\mathcal{A}$, on input of base group parameters pp_G, always guarantees that for all $i \in [k]$, $[\mathbf{b}^{(0)}]_G, [\mathbf{b}^{(1)}]_G \in \mathcal{B}^n_{pp_G}$ and $\boldsymbol{\nu}^{(0),(i)}, \boldsymbol{\nu}^{(1),(i)} \in \mathbb{Z}_p^n$ and $[x_i^*]_G := \left[(\boldsymbol{\nu}^{(0),(i)})^\intercal \cdot \mathbf{b}^{(0)}\right]_G = \left[(\boldsymbol{\nu}^{(1),(i)})^\intercal \cdot \mathbf{b}^{(1)}\right]_G$. We proceed over a series of hybrids as defined in Figure 12.7.

GAME $\mathbf{G}^b_0$. This game is the original switching game $\mathrm{Exp}^{\text{k-switching}}_{\mathbb{H},\mathcal{A},b}(\lambda)$ as defined in Figure 11.1a.

GAME $\mathbf{G}^b_1$. The difference between $\mathbf{G}^b_1$ and $\mathbf{G}^b_0$ is only conceptual. The only difference is that $[x_i^*]_G$ is computed as $\left[(\mathbf{b}^{(0)})^\intercal \cdot \boldsymbol{\nu}^{(0),(i)}\right]_G$ instead of $\left[(\mathbf{b}^{(b)})^\intercal \cdot \boldsymbol{\nu}^{(b),(i)}\right]_G$. Since $\mathcal{A}$ is legitimate, $[x_i^*]_G = \left[(\mathbf{b}^{(0)})^\intercal \cdot \boldsymbol{\nu}^{(0),(i)}\right]_G = \left[(\mathbf{b}^{(1)})^\intercal \cdot \boldsymbol{\nu}^{(1),(i)}\right]_G$, $[x_i^*]_G$ is independent of b. Hence, $\Pr[out_1^b = 1] = \Pr[out_0^b = 1]$.

GAME $\mathbf{G}^b_2$. The only difference between the games $\mathbf{G}^b_1$ and $\mathbf{G}^b_2$ is that $pp_{\mathbb{H}}$ is produced via $\mathsf{GGen}_{\mathbb{H}}(pp_G, [\mathbf{b}^{(b)}]_G)$ and $\mathsf{GGen}'_{\mathbb{H}}(pp_G, [\mathbf{b}^{(b)}]_G)$, respectively. Let $\mathcal{A}$ be a distinguisher between $\mathbf{G}^b_1$ and $\mathbf{G}^b_2$. We construct an adversary $\mathcal{B}^b_2$ against the property of Lemma 12.4. On input of pp_G, $\mathcal{B}^b_2$ calls $\mathcal{A}$ on input of pp_G to obtain $([\mathbf{b}^{(0)}]_G, [\mathbf{b}^{(1)}]_G, (\boldsymbol{\nu}^{(0),(i)})_{i\in[k]}, (\boldsymbol{\nu}^{(1),(i)})_{i\in[k]}, st)$. $\mathcal{B}^b_2$ outputs $([\mathbf{b}^{(b)}]_G, st)$ and obtains $(pp_{\mathbb{H}}, \tau_{\mathbb{H}})$ which is either sampled according to $\mathsf{GGen}_{\mathbb{H}}(pp_G, [\mathbf{b}^{(b)}]_G)$ or according to $\mathsf{GGen}'_{\mathbb{H}}(pp_G, [\mathbf{b}^{(b)}]_G)$. Since, $\tau_{\mathbb{H}}$ contains sk and $(\alpha_1, \ldots, \alpha_n)$, $\mathcal{B}^b_2$ is able to simulate the game $\mathbf{G}^b_1$ (respectively, $\mathbf{G}^b_2$) for $\mathcal{A}$. More precisely, $\mathcal{B}^b_2$ samples $\widehat{h_i^*}$ as in $\mathbf{G}^b_1$, calls $\mathcal{A}$ on input of $(pp_{\mathbb{H}}, (\widehat{h_i^*})_{i\in[k]})$ and outputs $\mathcal{A}$'s output. If $pp_{\mathbb{H}}$ is sampled using $\mathsf{GGen}_{\mathbb{H}}$, $\mathcal{B}^b_2$ perfectly simulates $\mathbf{G}^b_1$ for $\mathcal{A}$ and if $pp_{\mathbb{H}}$ is sampled using $\mathsf{GGen}'_{\mathbb{H}}$, $\mathcal{B}^b_2$ perfectly simulates $\mathbf{G}^b_2$ for $\mathcal{A}$. Hence,

$$\left|\Pr\left[out_2^b = 1\right] - \Pr\left[out_1^b = 1\right]\right| \leqslant \mathrm{Adv}^{\mathsf{swap}}_{\mathbb{H},\mathcal{B}^b_2}(\lambda).$$

GAME $\mathbf{G}^b_3$. $\mathbf{G}^b_3$ is identical to $\mathbf{G}^b_2$ but instead of using the witnesses $(sk, \boldsymbol{\nu}_i^*)$ to compute the consistency proofs π_i^*, $\mathbf{G}^b_3$ uses w_y as witness for all consistency

$\mathbf{G}_1^b$

$pp_G \leftarrow \mathsf{GGen}_G(1^\lambda)$
$(([\mathbf{b}^{(j)}]_G)_j, (\boldsymbol{\nu}^{(j),(i)})_{i,j}) \leftarrow \mathcal{A}(1^\lambda, pp_G)$
$(pp_{\mathbb{H}}, \tau_{\mathbb{H}}) \leftarrow \mathsf{GGen}_{\mathbb{H}}(pp_G, [\mathbf{b}^{(b)}]_G)$
$[x_i^*]_G := [(\mathbf{b}^{(0)})^\intercal \cdot \boldsymbol{\nu}^{(0),(i)}]_G$
$\boldsymbol{\nu}_i^* := \left(\nu_1^{(b),(i)} \cdot \alpha_1^{-1}, \ldots, \nu_n^{(b),(i)} \cdot \alpha_n^{-1}\right)^\intercal$
$C_i^* := \mathsf{Enc}(pk, \boldsymbol{\nu}_i^*)$
$\pi_i^* \leftarrow \mathsf{Prove}(\sigma, (pars, [x_i^*]_G, C_i^*), (sk, \boldsymbol{\nu}_i^*))$
$\widehat{h_i^*} := ([x_i^*]_G, C_i^*, \pi_i^*)_{\mathbb{H}}$
return $\mathcal{A}(pp_{\mathbb{H}}, (\widehat{h_i^*})_{i\in[k]})$

(a) Description of $\mathbf{G}_1^b$.

$\mathbf{G}_2^b$

$pp_G \leftarrow \mathsf{GGen}_G(1^\lambda)$
$(([\mathbf{b}^{(j)}]_G)_j, (\boldsymbol{\nu}^{(j),(i)})_{i,j}) \leftarrow \mathcal{A}(1^\lambda, pp_G)$
$(pp_{\mathbb{H}}, \tau_{\mathbb{H}}) \leftarrow \mathsf{GGen}'_{\mathbb{H}}(pp_G, [\mathbf{b}^{(b)}]_G)$
$[x_i^*]_G := [(\mathbf{b}^{(0)})^\intercal \cdot \boldsymbol{\nu}^{(0),(i)}]_G$
$\boldsymbol{\nu}_i^* := \left(\nu_1^{(b),(i)} \cdot \alpha_1^{-1}, \ldots, \nu_n^{(b),(i)} \cdot \alpha_n^{-1}\right)^\intercal$
$C_i^* := \mathsf{Enc}(pk, \boldsymbol{\nu}_i^*)$
$\pi_i^* \leftarrow \mathsf{Prove}(\sigma, (pars, [x_i^*]_G, C_i^*), (sk, \boldsymbol{\nu}_i^*))$
$\widehat{h_i^*} := ([x_i^*]_G, C_i^*, \pi_i^*)_{\mathbb{H}}$
return $\mathcal{A}(pp_{\mathbb{H}}, (\widehat{h_i^*})_{i\in[k]})$

(b) Description of $\mathbf{G}_2^b$.

$\mathbf{G}_3^b$

$pp_G \leftarrow \mathsf{GGen}_G(1^\lambda)$
$(([\mathbf{b}^{(j)}]_G)_j, (\boldsymbol{\nu}^{(j),(i)})_{i,j}) \leftarrow \mathcal{A}(1^\lambda, pp_G)$
$(pp_{\mathbb{H}}, \tau_{\mathbb{H}}) \leftarrow \mathsf{GGen}'_{\mathbb{H}}(pp_G, [\mathbf{b}^{(b)}]_G)$
$[x_i^*]_G := [(\mathbf{b}^{(0)})^\intercal \cdot \boldsymbol{\nu}^{(0),(i)}]_G$
$\boldsymbol{\nu}_i^* := \left(\nu_1^{(b),(i)} \cdot \alpha_1^{-1}, \ldots, \nu_n^{(b),(i)} \cdot \alpha_n^{-1}\right)^\intercal$
$C_i^* := \mathsf{Enc}(pk, \boldsymbol{\nu}_i^*)$
$\pi_i^* \leftarrow \mathsf{Prove}(\sigma, (pars, [x_i^*]_G, C_i^*), (w_y))$
$\widehat{h_i^*} := ([x_i^*]_G, C_i^*, \pi_i^*)_{\mathbb{H}}$
return $\mathcal{A}(pp_{\mathbb{H}}, (\widehat{h_i^*})_{i\in[k]})$

(c) Description of $\mathbf{G}_3^b$.

$\mathbf{G}_4^b$

$pp_G \leftarrow \mathsf{GGen}_G(1^\lambda)$
$(([\mathbf{b}^{(j)}]_G)_j, (\boldsymbol{\nu}^{(j),(i)})_{i,j}) \leftarrow \mathcal{A}(1^\lambda, pp_G)$
$(pp_{\mathbb{H}}, \tau_{\mathbb{H}}) \leftarrow \mathsf{GGen}'_{\mathbb{H}}(pp_G, [\mathbf{b}^{(b)}]_G)$
$[x_i^*]_G := [(\mathbf{b}^{(0)})^\intercal \cdot \boldsymbol{\nu}^{(0),(i)}]_G$
$\boldsymbol{\nu}_i^* := (0, \ldots, 0)^\intercal$
$C_i^* := \mathsf{Enc}(pk, \boldsymbol{\nu}_i^*)$
$\pi_i^* \leftarrow \mathsf{Prove}(\sigma, (pars, [x_i^*]_G, C_i^*), (w_y))$
$\widehat{h_i^*} := ([x_i^*]_G, C_i^*, \pi_i^*)_{\mathbb{H}}$
return $\mathcal{A}(pp_{\mathbb{H}}, (\widehat{h_i^*})_{i\in[k]})$

(d) Description of $\mathbf{G}_4^b$.

Figure 12.7: Games used in proof of Lemma 12.5. Recall that $\alpha_1 = 1$ and $\alpha_2, \ldots, \alpha_n$ are uniformly distributed over $\mathbb{Z}_p^\times$.

proofs. Since σ is produced via $\mathsf{HSetup}(1^\lambda)$, the proofs produced in $\mathbf{G}_3^b$ and $\mathbf{G}_2^b$ are *identically* distributed due to the perfect witness indistinguishability of NIWI. Hence, $\Pr[out_2^b = 1] = \Pr[out_3^b = 1]$.

GAME $\mathbf{G}_4^b$. $\mathbf{G}_4^b$ is identical to $\mathbf{G}_3^b$ except for the fact that $\mathbf{G}_4^b$ always uses the zero vector $(0, \ldots, 0)^\intercal$ as $\boldsymbol{\nu}_i^*$ instead of $\boldsymbol{\nu}_i^* := (\nu_1^{(b),(i)} \cdot \alpha_1^{-1}, \ldots, \nu_n^{(b),(i)} \cdot \alpha_n^{-1})^\intercal$. Note that the secret decryption key sk is not necessary to simulate $\mathbf{G}_3^b$ and $\mathbf{G}_4^b$. Additionally, $\boldsymbol{\nu}_i^*$ is only used to call $\mathsf{Enc}(pk, \boldsymbol{\nu}_i^*)$ (in $\mathbf{G}_3^b$). This allows to apply the IND-CPA security of FHE.

CLAIM 12.1. For all legitimate k-switching PPT adversaries $\mathcal{A}$, there exists a (legitimate) PPT adversary $\mathcal{B}_4^b$ such that

$$\left|\Pr\left[out_4^b = 1\right] - \Pr\left[out_3^b = 1\right]\right| \leqslant k \cdot \mathrm{Adv}^{\text{ind-cpa}}_{\mathsf{FHE},\mathcal{B}_4^b}(\lambda).$$

Proof of Claim 12.1. We use a standard hybrid argument.

GAME $\mathbf{G}_{3.1}^b$. The hybrid game $\mathbf{G}_{3.1}^b$ is defined as $\mathbf{G}_3^b$ except that for $j < i$, C_j^* is produced via $\mathsf{Enc}(pk, 0)$ and for $j \geqslant i$, C_j^* is produced for $\mathsf{Enc}(pk, \boldsymbol{\nu}_j^*)$. Hence, $\mathbf{G}_{3.1}^b$ is identical to $\mathbf{G}_3^b$ and $\mathbf{G}_{3.k+1}^b$ is identical to $\mathbf{G}_4^b$. We construct an adversary $\mathcal{B}_{3.1}^b$ for the IND-CPA security of FHE. On input of pk, $\mathcal{B}_{3.1}^b$ guesses

an index $\nu \in [k]$, samples $pp_{\mathbb{G}} \leftarrow \mathsf{GGen}_{\mathbb{G}}(1^\lambda)$, calls $\mathcal{A}(pp_{\mathbb{G}})$, obtains $\mathcal{A}$'s output $(([\mathbf{b}^{(j)}]_{\mathbb{G}})_{j\in\{0,1\}}, (\boldsymbol{\nu}^{(j),(i)})_{i\in[k],j\in\{0,1\}})$ and produces $pp_{\mathbb{H}}$ as in $\mathbf{G}_3^b$. Note that $\mathcal{B}_{3.1}^b$ knows $\alpha_1,\dots,\alpha_n$ and w_y. For $\mu \in \{1,\dots,\nu-1\}$, $\mathcal{B}_{3.1}^b$ produces $\widehat{h_\mu^*}$ as in $\mathbf{G}_3^b$. For $\mu \in \{\nu+1,\dots,k\}$, $\mathcal{B}_{3.1}^b$ produces $\widehat{h_\mu^*}$ as in $\mathbf{G}_4^b$. For $\mu = \nu$, $\mathcal{B}_{3.1}^b$ outputs

$$M_0 \coloneqq \left(\nu_1^{(b),(\mu)} \cdot \alpha_1^{-1}, \dots, \nu_n^{(b),(\mu)} \cdot \alpha_n^{-1}\right)^\intercal$$
$$M_1 \coloneqq (0,\dots,0)^\intercal$$

to the IND-CPA game together with his internal state. On input of C^*, $\mathcal{B}_{3.1}^b$ uses C^* as C_μ^*, computes

$$[x_\mu^*]_{\mathbb{G}} \coloneqq \left[(\mathbf{b}^{(0)})^\intercal \cdot \boldsymbol{\nu}^{(0),(\mu)}\right]_{\mathbb{G}},$$
$$\pi_\mu^* \leftarrow \mathsf{Prove}\left(\sigma, (pars, [x_\mu^*]_{\mathbb{G}}, C_\mu^*), (w_y)\right).$$

Finally, $\mathcal{B}_{3.1}^b$ calls $\mathcal{A}$ on input $(pp_{\mathbb{H}}, (\widehat{h_i^*})_{i\in[k]})$ and outputs $\mathcal{A}$'s output. If C^* is an encryption of M_0, $\mathcal{B}_{3.1}^b$ perfectly simulates $\mathbf{G}_{3.\nu}^b$, otherwise $\mathcal{B}_{3.1}^b$ perfectly simulates $\mathbf{G}_{3.\nu+1}^b$. Hence,

$$\left|\Pr\left[out_4^b = 1\right] - \Pr\left[out_3^b = 1\right]\right| \leqslant k \cdot \mathrm{Adv}^{\mathsf{ind\text{-}cpa}}_{\mathsf{FHE},\mathcal{B}_{3.1}^b}(\lambda).$$

□

Recall that every legitimate k-switching adversary always guarantees

$$\left[(\boldsymbol{\nu}^{(0),(i)})^\intercal \cdot \mathbf{b}^{(0)}\right]_{\mathbb{G}} = \left[(\boldsymbol{\nu}^{(1),(i)})^\intercal \cdot \mathbf{b}^{(1)}\right]_{\mathbb{G}}$$

and $[\mathbf{b}^{(0)}]_{\mathbb{G}}, [\mathbf{b}^{(1)}]_{\mathbb{G}} \in \mathcal{B}_{pp_{\mathbb{G}}}^n$. Therefore, $\mathbf{G}_4^0$ and $\mathbf{G}_4^1$ only differ in the fact that in $\mathbf{G}_4^b$ the vector $[\mathbf{b}^{(b)}]_{\mathbb{G}}$ is used to compute $[\boldsymbol{\Omega}]_{\mathbb{G}}$. That is, $[\boldsymbol{\Omega}]_{\mathbb{G}} = ([1]_{\mathbb{G}}, [b_{2,b}]_{\mathbb{G}}^{\alpha_2}, \dots, [b_{n,b}]_{\mathbb{G}}^{\alpha_n})^\intercal$ for $\alpha_2,\dots,\alpha_n$ chosen uniformly at random from $\mathbb{Z}_p^n$. Except for $[\boldsymbol{\Omega}]_{\mathbb{G}}$, the view of $\mathcal{A}$ is independent of $\alpha_2,\dots,\alpha_n$.[49] Thus, $\Pr[out_4^0 = 1] = \Pr[out_4^1 = 1]$.

[49] *Note that since* $[\mathbf{b}^{(0)}]_{\mathbb{G}}, [\mathbf{b}^{(1)}]_{\mathbb{G}} \in \mathcal{B}_{pp_{\mathbb{G}}}^n$, *we have* $[b_{i,b}]_{\mathbb{G}} \neq [0]_{\mathbb{G}}$ *(for all* $i \in n$*).*

Note that since $\mathbb{G}$ has unique encodings, $\mathcal{A}$ is unable to extract auxiliary information from the encodings of $[x_i^*]_{\mathbb{G}}$. This is crucial since such auxiliary information may for instance reveal how $[x_i^*]_{\mathbb{G}}$ was computed. □

LEMMA 12.6. *The group scheme* $\mathbb{H}$ *defined in Figures 12.1 and 12.2 satisfies statistical re-randomizability.*

Proof of Lemma 12.6. The circuit C_{rerand} takes inputs from $\{0,1\}^{p_{enc}(\lambda)}$ and expects randomness from $\{0,1\}^{m'(\lambda)} \times \{0,1\}^{m''(\lambda)}$. We recall that $\mathsf{piO}_\ell^\star$ is an ℓ-expanding pIO scheme for $\ell(\lambda) = m'(\lambda) + m''(\lambda) + 2(\lambda+1) + 3$. Since for every distribution Z_1 over $\{0,1\}^{p_{enc}(\lambda)}$,

$$\widetilde{H}_\infty(U_{\ell(\lambda)} \mid Z_1) = \ell(\lambda) > m'(\lambda) + m''(\lambda) + 2(\lambda+1) + 2,$$

the statistical distance

$$\Delta\Big(\{\Lambda_{\mathsf{rerand}} \leftarrow \mathsf{piO}_\ell^\star(C_{\mathsf{rerand}}) \colon (\Lambda_{\mathsf{rerand}}, \Lambda_{\mathsf{rerand}}(Z_1, Z_2))\},$$
$$\{\Lambda_{\mathsf{rerand}} \leftarrow \mathsf{piO}_\ell^\star(C_{\mathsf{rerand}}) \colon (\Lambda_{\mathsf{rerand}}, C_{\mathsf{rerand}}(Z_1; U_{m'(\lambda)+m''(\lambda)}))\}\Big) \tag{12.1}$$

is at most $2^{-(\lambda+1)}$.

Let $\widehat{h_0} =: ([x_0]_G, C_0, \pi_0)_{\mathbb{H}}, \widehat{h_1} =: ([x_1]_G, C_1, \pi_1)_{\mathbb{H}} \in \mathbb{H}$ be the encodings chosen by the adversary $\mathcal{A}$. Since $\mathcal{A}$ is a legitimate re-randomization adversary, $\mathsf{PrivExt}_{\mathbb{H}}(\tau_{\mathbb{H}}, \widehat{h_0}) = \mathsf{PrivExt}_{\mathbb{H}}(\tau_{\mathbb{H}}, \widehat{h_1})$. Due to perfect correctness of FHE and since $\alpha_1, \ldots, \alpha_n \in \mathbb{Z}_p^\times$ are invertible, $\mathsf{Dec}(sk, C_0) = \mathsf{Dec}(sk, C_1)$. Due to perfect re-randomizability of FHE, the ciphertexts produced via $C_{\mathsf{rerand}}(\widehat{h_0})$ and $C_{\mathsf{rerand}}(\widehat{h_1})$ are identically distributed. Furthermore, since $C_{\mathsf{rerand}}(\widehat{h_b})$ produces the consistency proof using the witness $(sk, \mathsf{Dec}(sk, C_b))$, the statistical distance

$$\Delta\Big(C_{\mathsf{rerand}}\Big(\widehat{h_0}; U_{m'(\lambda)+m''(\lambda)}\Big), C_{\mathsf{rerand}}\Big(\widehat{h_1}; U_{m'(\lambda)+m''(\lambda)}\Big)\Big) = 0. \tag{12.2}$$

Therefore, combining Equations (12.1) and (12.2) yields

$$\mathrm{Adv}^{\mathsf{rerand}}_{\mathbb{H},\mathcal{A}}(\lambda) \leqslant 2 \cdot 2^{-(\lambda+1)} = 2^{-\lambda}.$$

Note that since $\mathbb{G}$ has unique encodings, $\mathcal{A}$ is unable to extract auxiliary information from the encodings of $\mathsf{Unwrap}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{h})$. This is crucial since such auxiliary information may be used to distinguish whether $\widehat{h_0}$ or $\widehat{h_1}$ was used to derive $\widehat{h}$. □

Theorem 12.1 then follows by combining Lemmas 12.1, 12.2, 12.3, 12.5 and 12.6. □

13 SIGNED ELGAMAL

The ElGamal public-key encryption scheme [ElG85] is a classical public-key encryption scheme based on cyclic groups. Given some cyclic group $\mathbb{G}$ of prime order with generator g, a public-key secret-key pair is of the form (g^y, y) for $y \in \mathbb{Z}_p$. Encryption of some message $M \in \mathbb{G}$ chooses a random $x \leftarrow \mathbb{Z}_p$ and outputs the ciphertext $(C_1, C_2) := (g^x, g^x \cdot M)$. Decryption recovers M by computing $C_2 \cdot (C_1)^{-y}$. The ElGamal encryption scheme is IND-CPA secure under the DDH assumption in $\mathbb{G}$, [TY98]. That is, no adversary can distinguish encryptions of two messages, even if the public key is known. The de-facto security notion for public-key encryption is IND-CCA2 security introduced in [RS92]. IND-CCA2 security requires that no adversary is able to distinguish encryptions of two messages even when granted the ability to decrypt arbitrary ciphertexts during the entire experiment (even after given the challenge ciphertext), as long as he does not use this ability to decrypt the challenge ciphertext. The ElGamal encryption scheme is homomorphic and can hence not be IND-CCA2 secure. As detailed in [FPS20], the ElGamal encryption scheme can be made IND-CCA2 secure by making it plaintext-aware [BR95] or by considering its hashed variant [ABR01].

One possibility of making ElGamal plaintext-aware is to use a Schnorr signature with g^x as public key. The resulting scheme is called Schnorr-signed ElGamal [Jak98; TY98]. [SJ00] show that Schnorr-signed ElGamal is IND-CCA2 secure in the generic group model and the random oracle model and [TY98] prove Schnorr-signed ElGamal IND-CCA2 secure under some non-standard knowledge-type assumptions on Schnorr signatures.

The hashed variant of the ElGamal encryption scheme derives the key via $H((g^y)^x)$ (instead of $(g^y)^x$). More precisely, in the hashed ElGamal key-encapsulation mechanism (KEM), a public key is a group element Y, the corresponding secret key is $y = dlog_g(Y)$. For encryption, one picks a random exponent $x \leftarrow \mathbb{Z}_p$ to compute a key $H(Y^x)$ accompanied by an encapsulation $X := g^x$. Given the encapsulation and the secret key y, the receiver can recover that key $K = H(X^y)$. In the random oracle model the hashed ElGamal key encapsulation algorithm (KEM) is IND-CCA2 secure under the gap Diffie-Hellman assumption[50] [CS03].

[FPS20] showed that a combination of Schnorr-signed ElGamal and hashed ElGamal (henceforth simply called Schnorr-signed ElGamal) is tightly[51] IND-CCA2 secure under the DL assumption in the algebraic group model and the random oracle model. Schnorr-signed ElGamal (see Figure 13.1) works similarly as hashed ElGamal but every encapsulation is accompanied by a Schnorr signature for message X under public key X. Decryption works as before with the difference that decryption aborts if the provided Schnorr signature is invalid.

In this chapter, we demonstrate that the algebraic wrapper can be applied to mimic the proof of tight IND-CCA2 security of Schnorr-signed ElGamal $\mathsf{E}_{\mathsf{sElG}}$ from [FPS20]. Note that in contrast to the tight reduction for Schnorr

[50] *The gap Diffie-Hellman assumption states that the computational Diffie-Hellman problem is hard even in the presence of a decisional Diffie-Hellman oracle.*

[51] *A cryptographic building block is tightly secure under some assumption A if every adversary breaking the building block can be transformed into a problem solver solving A with roughly the same runtime and success probability as the adversary.*

$\mathsf{KGen}(pp_{\mathbb{H}})$

$y \leftarrow \mathbb{Z}_p$

$\widehat{G} \leftarrow \mathsf{Rerand}_{\mathbb{H}}(pp_{\mathbb{H}}, \mathsf{Sam}_{\mathbb{H}}(pp_{\mathbb{H}}, 1))$

$\widehat{Y} \leftarrow \mathsf{Rerand}_{\mathbb{H}}(pp_{\mathbb{H}}, \mathsf{Sam}_{\mathbb{H}}(pp_{\mathbb{H}}, y))$

$pk \coloneqq (pp_{\mathbb{H}}, \widehat{G}, \widehat{Y})$

$sk \coloneqq (pk, y)$

return (pk, sk)

(a) The key generation algorithm.

$\mathsf{Enc}(pk = (pp_{\mathbb{H}}, \widehat{G}, \widehat{Y}))$

$x, r \leftarrow \mathbb{Z}_p$

$\widehat{R} \leftarrow \mathsf{Rerand}_{\mathbb{H}}(pp_{\mathbb{H}}, \mathsf{Sam}_{\mathbb{H}}(pp_{\mathbb{H}}, r))$

$\widehat{X} \leftarrow \mathsf{Rerand}_{\mathbb{H}}(pp_{\mathbb{H}}, \mathsf{Sam}_{\mathbb{H}}(pp_{\mathbb{H}}, x))$

$k \coloneqq H'(\widehat{Y}^{x})$

$s \coloneqq r + H(\widehat{R}, \widehat{X}) \cdot x \mod p$

return $(k, (\widehat{X}, \widehat{R}, s))$

(b) The encapsulation algorithm.

$\mathsf{Dec}(sk, (\widehat{X}, \widehat{R}, s))$

if $[s]_{\mathbb{H}} \neq_{\mathbb{H}} \widehat{X}^{H(\widehat{R},\widehat{X})} \cdot \widehat{R}$ **then**

  return $\perp$

return $k \coloneqq H'(\widehat{X}^{y})$

(c) The decapsulation algorithm.

Figure 13.1: The Schnorr-signed ElGamal encryption scheme $\mathsf{E}_{\mathsf{sElG}}$ instantiated with an algebraic wrapper $\mathbb{H}$. Note that to compensate for the non-uniqueness of group element encodings, the (random oracle) hash value of a group element encoding is computed for the identifier produced by $\mathsf{GetID}_{\mathbb{H}}(pp_{\mathbb{H}}, \cdot)$ (which is unique for all group element encodings of the same group element). The hash function H maps tuples of group elements from $\mathbb{H}$ to elements in $\mathcal{K}$ and the hash function H' maps group elements from $\mathbb{H}$ to $\mathbb{Z}_p$ elements.

signatures from [AHK20], the tightness for Schnorr-signed ElGamal does not require the "origin element trick" in combination with statistical re-randomizability since it is not necessary to switch the representations of oracle responses. Note that the "origin element trick" is not part of this thesis.

THEOREM 13.1. *Let* $\mathsf{GGen}_{\mathbb{G}}$ *be a group generator for a cyclic group* $\mathbb{G}$ *such that* DL *is hard relative to* $\mathsf{GGen}_{\mathbb{G}}$ *and let* $\mathbb{H}$ *be an algebraic wrapper for* $\mathbb{G}$*. Then,* $\mathsf{E}_{\mathsf{sElG}}$ *in* $\mathbb{H}$ *is tightly* IND-CCA2 *secure in the random oracle model.*

More precisely, for all PPT *adversaries* $\mathcal{A}$*, there exists a* PPT *adversary* $\mathcal{B}$ *and a legitimate switching adversary* $\mathcal{A}'$ *both running in time* $T(\mathcal{B}) \approx T(\mathcal{A}) + (q_d + q_h) \cdot \mathit{poly}(\lambda)$ *and* $T(\mathcal{A}') \approx T(\mathcal{A}) + (q_d + q_h) \cdot \mathit{poly}(\lambda)$ *such that*

$$\mathrm{Adv}^{\text{ind-cca2}}_{\mathsf{E}_{\mathsf{sElG}},\mathcal{A}}(\lambda) \leqslant \mathrm{Adv}^{\mathsf{dl}}_{\mathcal{B},\mathbb{G}}(\lambda) + \mathrm{Adv}^{\text{2-switching}}_{\mathcal{A}',\mathbb{H}}(\lambda) + \frac{\mathcal{O}(q_d + q_h)}{2^{\lambda}},$$

where q_h *is a polynomial upper bound on the number of random oracle queries,* q_d *is a polynomial upper bound on the number of decryption queries and poly is a polynomial independent of* q_d *and* q_h*.*

Proof. The proof strategy follows (up to some preparations) the outline of [FPS20]. The hybrid $\mathbf{G}_0$ is identical to $\mathrm{Exp}^{\text{ind-cca2}}_{\mathsf{E}_{\mathsf{sElG}},\mathcal{A}}(\lambda)$. The initial game transitions until hybrid $\mathbf{G}_3$ are preparation steps due to the algebraic wrapper. The following hybrids $\mathbf{G}_4$, $\mathbf{G}_5$ and $\mathbf{G}_6$ correspond exactly to the hybrids G_1, G_2, G_3 used in [FPS20], respectively. The preparation steps set up the randomness for the challenge ciphertext as $x^* \coloneqq z \cdot y$. Further, the randomness for

the signature in the challenge ciphertext is chosen using an x^*-component. Subsequently, re-randomizability and switching are applied such that the public key $\widehat{Y}$ uses the representation vector $(0,1)^\intercal$ and the randomness for the challenge ciphertext $\widehat{X^*}$ uses the representation vector $(0,z)^\intercal$. The remaining proof proceeds as in [FPS20].

For notational convenience, we introduce the notation $\widehat{A} =_{\mathbb{H}} \widehat{B}$ for the equality testing algorithm $\mathsf{Eq}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{A}, \widehat{B})$. Note that the hash values for a group element encoding $\widehat{A}$ are produced for the bitstring derived via $\mathsf{GetID}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{A})$ which is guaranteed to be unique for all group element encodings $\widehat{B}$ satisfying $\widehat{A} =_{\mathbb{H}} \widehat{B}$.

We proceed over a series of games starting from the IND-CCA2 game in the random oracle model. The hash functions $\widetilde{H}\colon \mathbb{H} \times \mathbb{H} \to \mathbb{Z}_p$ and $\widetilde{H}'\colon \mathbb{H} \to \mathcal{K}$ behave exactly as there counterparts H and H', respectively, and act solely as helper functions. The adversary $\mathcal{A}$ only has access to the oracles H and H' (and Dec). Throughout the proof, the behavior of $\widetilde{H}$ and $\widetilde{H}'$ will *not* be altered.

GAME $\mathbf{G}_0$. $\mathbf{G}_0$ is identical to the IND-CCA2 security game in the random oracle model, see Figure 13.2.

GAME $\mathbf{G}_1$. We first change how the signature in the challenge ciphertext is generated. Particularly, the randomness used for the signature is chosen using a y-component. More precisely, $\mathbf{G}_1$ (defined in Figure 13.3) is identical to $\mathbf{G}_0$ except for two key differences. First, x^* is not sampled directly from $\mathbb{Z}_p$ but is computed as $z \cdot y$ for some uniformly chosen $z \leftarrow \mathbb{Z}_p$. Second, in $\mathbf{G}_0$, the first part of the signature $\widehat{R^*}$ is produced via $\mathsf{Rerand}_{\mathbb{H}}(pp_{\mathbb{H}}, \mathsf{Sam}_{\mathbb{H}}(pp_{\mathbb{H}}, x^*))$ and is used to obtain the hash value $c^* := \widetilde{H}(\widehat{R^*}, \widehat{X^*})$ and is itself part of the signature. In $\mathbf{G}_1$, the group element which is used to obtain the hash value c^* and the group element which is actually part of the signature are decoupled. To obtain c^*, $\mathbf{G}_1$ uses $\widehat{R_1^*}$ which is produced as in $\mathbf{G}_0$ via $\mathsf{Rerand}_{\mathbb{H}}(pp_{\mathbb{H}}, \mathsf{Sam}_{\mathbb{H}}(pp_{\mathbb{H}}, x^*))$. The group element $\widehat{R_2^*}$ which is part of the signature is computed later, depending on c^*, via $\mathsf{Rerand}_{\mathbb{H}}(pp_{\mathbb{H}}, \mathsf{Sam}_{\mathbb{H}}(pp_{\mathbb{H}}, s^* - c^* \cdot x^*))$.

Since x^* is in both games uniformly distributed and $r^* = s^* - c^* \cdot x^* \bmod p$ and thus

$$\mathsf{GetID}_{\mathbb{H}}\left(pp_{\mathbb{H}}, \widehat{R_1^*}\right) = \mathsf{GetID}_{\mathbb{H}}\left(pp_{\mathbb{H}}, \widehat{R_2^*}\right),$$

the view of $\mathcal{A}$ in $\mathbf{G}_0$ and $\mathbf{G}_1$ is identical and $\Pr[out_0 = 1] = \Pr[out_1 = 1]$.

GAME $\mathbf{G}_2$. $\mathbf{G}_2$ (see Figure 13.4) is identical to $\mathbf{G}_1$ except that the encodings $\widehat{Y}$, $\widehat{X^*}$ and $\widehat{R_2^*}$ are produced via private sampling or via the group operation instead of public sampling. Since these encodings are re-randomized subsequently, this game hop is justified by the re-randomizability of the algebraic wrapper $\mathbb{H}$. More precisely, we successively replace

$$\begin{aligned}
&\mathsf{Rerand}_{\mathbb{H}}\left(pp_{\mathbb{H}}, \mathsf{Sam}_{\mathbb{H}}(pp_{\mathbb{H}}, y)\right) \text{ by } \mathsf{Rerand}_{\mathbb{H}}\left(pp_{\mathbb{H}}, \mathsf{PrivSam}_{\mathbb{H}}(\tau_{\mathbb{H}}, y)\right),\\
&\mathsf{Rerand}_{\mathbb{H}}\left(pp_{\mathbb{H}}, \mathsf{Sam}_{\mathbb{H}}(pp_{\mathbb{H}}, x^*)\right) \text{ by } \mathsf{Rerand}_{\mathbb{H}}\left(pp_{\mathbb{H}}, \mathsf{PrivSam}_{\mathbb{H}}(\tau_{\mathbb{H}}, x^*)\right),\\
&\mathsf{Rerand}_{\mathbb{H}}\left(pp_{\mathbb{H}}, \mathsf{Sam}_{\mathbb{H}}(pp_{\mathbb{H}}, s^* - c^* \cdot x^*)\right) \text{ by}\\
&\qquad \mathsf{Rerand}_{\mathbb{H}}\left(pp_{\mathbb{H}}, \widehat{G}^{s^*} \cdot (\widehat{X^*})^{-c^*}\right).
\end{aligned}$$

Due to correctness of sampling, we have

$$\begin{aligned}\mathsf{PrivExt}_{\mathbb{H}}\left(\tau_{\mathbb{H}}, \mathsf{Sam}_{\mathbb{H}}(pp_{\mathbb{H}}, y)\right) &= y \cdot \boldsymbol{e_1} \\ &= \mathsf{PrivExt}_{\mathbb{H}}\left(\tau_{\mathbb{H}}, \mathsf{PrivSam}_{\mathbb{H}}(\tau_{\mathbb{H}}, y)\right), \\ \mathsf{PrivExt}_{\mathbb{H}}\left(\tau_{\mathbb{H}}, \mathsf{Sam}_{\mathbb{H}}(pp_{\mathbb{H}}, x^*)\right) &= x^* \cdot \boldsymbol{e_1} \\ &= \mathsf{PrivExt}_{\mathbb{H}}\left(\tau_{\mathbb{H}}, \mathsf{PrivSam}_{\mathbb{H}}(\tau_{\mathbb{H}}, x^*)\right).\end{aligned}$$

Further, due to correctness of sampling and correctness of extraction,

$$\begin{aligned}&\mathsf{PrivExt}_{\mathbb{H}}\left(\tau_{\mathbb{H}}, \mathsf{Sam}_{\mathbb{H}}(pp_{\mathbb{H}}, s^* - c^* \cdot x^*)\right) \\ &= (s^* - c^* \cdot x^*) \cdot \boldsymbol{e_1} \\ &= \mathsf{PrivExt}_{\mathbb{H}}\left(\tau_{\mathbb{H}}, \widehat{G}^{\,s^*} \cdot (\widehat{X^*})^{-c^*}\right).\end{aligned}$$

Hence, for each of these changes we can apply statistical re-randomizability of $\mathbb{H}$,

$$\left|\Pr\left[out_2 = 1\right] - \Pr\left[out_1 = 1\right]\right| \leqslant \frac{3}{2^\lambda}.$$

GAME $\mathbf{G}_3$. Towards removing the necessity to know y for the simulation, we change the basis to be $[\mathbf{b}]_G := ([1]_G, [y]_G)^\intercal$ and adapt the representation vectors used for private sampling of $\widehat{Y}$ and $\widehat{X^*}$ accordingly. More precisely, $\mathbf{G}_3$ is identical to $\mathbf{G}_2$ except for the fact that instead of the basis $([1]_G, [\beta_2]_G)^\intercal$ for some arbitrary $\beta_2 \in \mathbb{Z}_p$, we use the basis $([1]_G, [y]_G)^\intercal$. Additionally, the encodings $\widehat{Y}$ and $\widehat{X^*}$ are privately sampled such that they carry the representation vectors $(0,1)^\intercal$ and $(0,z)^\intercal$, respectively. Figure 13.5 provides a formal definition of $\mathbf{G}_3$.

This game hop is justified by 2-switching of $\mathbb{H}$. We construct an adversary $\mathcal{B}_3$ on 2-switching as follows. Initially, on input of pp_G, $\mathcal{B}_3$ outputs two basis vectors

$$\left[\mathbf{b}^{(\mathbf{G}_2)}\right]_G := ([1]_G, [\beta_2]_G)^\intercal \text{ and } \left[\mathbf{b}^{(\mathbf{G}_3)}\right]_G := ([1]_G, [y]_G)^\intercal$$

and representation vectors

$$\begin{aligned}\boldsymbol{v}^{(1),(\mathbf{G}_2)} &:= (y, 0)^\intercal, \quad \boldsymbol{v}^{(2),(\mathbf{G}_2)} := (z \cdot y, 0)^\intercal \\ \boldsymbol{v}^{(1),(\mathbf{G}_3)} &:= (0, 1)^\intercal, \quad \boldsymbol{v}^{(2),(\mathbf{G}_3)} := (0, z)^\intercal.\end{aligned}$$

In return, $\mathcal{B}_3$ receives public parameters $pp_{\mathbb{H}}$ and two encodings $\widehat{C^{(1)}}$ and $\widehat{C^{(2)}}$. $\mathcal{B}_3$ computes $\widehat{Y} \leftarrow \mathsf{Rerand}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{C^{(1)}})$ and $\widehat{X^*} \leftarrow \mathsf{Rerand}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{C^{(2)}})$ and simulates the remaining game as in $\mathbf{G}_2$. Note that this is possible since $\tau_{\mathbb{H}}$ is not necessary. $\mathcal{B}_3$ simulates either $\mathbf{G}_2$ or $\mathbf{G}_3$ for $\mathcal{A}$ depending on the challenge provided by $\mathrm{Exp}^{\text{2-switching}}_{\mathbb{H},\mathcal{A},b}(\lambda)$. Hence,

$$\left|\Pr\left[out_3 = 1\right] - \Pr\left[out_2 = 1\right]\right| \leqslant \mathrm{Adv}^{\text{2-switching}}_{\mathbb{H},\mathcal{B}_3}(\lambda).$$

Since $\mathcal{B}_3$ is a legitimate adversary, $\mathrm{Adv}^{\text{2-switching}}_{\mathbb{H},\mathcal{B}_3}(\lambda)$ is negligible.

GAME $\mathbf{G}_4$. From this point on, we are able to closely follow the lines of [FPS20]. $\mathbf{G}_4$ (see Figure 13.6) is identical to $\mathbf{G}_3$ except for the following modifications. In $\mathbf{G}_4$, the oracle H stores the private extractions of the encodings used to

call H in a list U. Furthermore, the decryption oracle obtains representation vectors corresponding to the supplied encodings $\widehat{R}$ and $\widehat{X}$ by first looking for a matching entry in U and, if no such entry is present, by applying private extraction. Let (ν, μ) and (ν', μ') be the thus obtained representation vectors of $\widehat{R}$ and $\widehat{X}$, respectively. Game $\mathbf{G}_4$ additionally introduces an abort condition. If $\mu + \mu' \cdot c = 0$ and $\mu' \neq 0$, $\mathbf{G}_4$ aborts and outputs a random bit. The games $\mathbf{G}_3$ and $\mathbf{G}_4$ only differ if $\mathbf{G}_4$ aborts.

Note that all values in the table T are set in an adversarial call to either H or Dec, except for $c^* = \mathsf{T}[(\mathsf{GetID}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{R_2^*}), \mathsf{GetID}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{X^*}))]$ which is set using $\widetilde{\mathsf{H}}$ within the game.[52] If $\mathsf{Dec}(\widehat{R}, \widehat{X}, s) \neq \perp$, then $(\widehat{R}, \widehat{X}) \neq (\widehat{R_2^*}, \widehat{X^*})$ since otherwise $s = s^*$. Hence, the value $c = \mathsf{T}[(\mathsf{GetID}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{R}), \mathsf{GetID}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{X}))]$ is independent of (ν, μ, ν', μ'). Therefore, the probability that $\mathbf{G}_4$ aborts is upper bounded by the probability that c is chosen as $c = -\frac{\mu}{\mu'} \bmod p$ which can be upper bounded by $\frac{1}{p} \leqslant 2^{-\lambda}$. By a union bound, the probability that $\mathbf{G}_4$ aborts is upper bounded by $\frac{q_d}{2^\lambda}$. Since $\mathbf{G}_3$ and $\mathbf{G}_4$ behave identical unless $\mathbf{G}_4$ aborts, we have

$$\left|\Pr\left[out_4 = 1\right] - \Pr\left[out_3 = 1\right]\right| \leqslant \frac{q_d}{2^\lambda}.$$

[52] *Recall that* T *keeps track of previous oracle responses of* H.

GAME $\mathbf{G}_5$. $\mathbf{G}_5$, defined in Figure 13.7, is identical to $\mathbf{G}_4$ except for the following changes. Instead of sampling r^* and querying $\widetilde{\mathsf{H}}$ for c^* to obtain $s^* = r^* + c^* \cdot x^*$, $\mathbf{G}_5$ samples s^* and c^* independently and computes $\widehat{R_2^*} = \mathsf{Rerand}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{G}^{s^*} \cdot (\widehat{X^*})^{-c^*})$ as in $\mathbf{G}_4$. This behavior is identical to $\mathbf{G}_4$ except for the event that the tuple $(\widehat{R_2^*}, \widehat{X^*})$ already has an entry in T. If this event occurs, $\mathbf{G}_5$ aborts. Since $\widehat{R_2^*}$ and $\widehat{X^*}$ are uniformly random and T contains at most $q_d + q_h$ many entries after at most q_d Dec-queries and q_h H-queries, the probability that $\mathbf{G}_5$ aborts, but $\mathbf{G}_4$ does not, can be upper bounded by $\frac{q_d + q_h}{2^{2\lambda}}$. Hence,

$$\left|\Pr\left[out_5 = 1\right] - \Pr\left[out_4 = 1\right]\right| \leqslant \frac{q_d + q_h}{2^{2\lambda}}.$$

GAME $\mathbf{G}_6$. $\mathbf{G}_6$ (see Figure 13.8) introduces two further abort conditions $(\star)$ and $(\star\star)$. $\mathbf{G}_6$ aborts as per $(\star)$ if H' is called on $\widehat{K}$ such that $\widehat{K} = \widehat{X^*}^y$ and aborts as per $(\star\star)$ if $\mu + \mu' \cdot \widetilde{\mathsf{H}}(\widehat{R}, \widehat{X}) \neq 0 \bmod p$, where μ and μ' are the $[y]_G$-components in the decomposition extracted from $\widehat{R}$ and $\widehat{X}$, respectively.

As in [FPS20], we show that if $\mathbf{G}_6$ differs from $\mathbf{G}_5$, then we can solve discrete logarithms.

We construct an adversary $\mathcal{B}_6$ on the discrete logarithm problem. Given $(pp_G, [1]_G, [y]_G)$, $\mathcal{B}_6$ produces $(pp_{\mathbb{H}}, \tau_{\mathbb{H}}) \leftarrow \mathsf{GGen}_{\mathbb{H}}(pp_G, ([1]_G, [y]_G)^\intercal)$ and simulates $\mathbf{G}_6$ for $\mathcal{A}$. Note that $\widehat{Y}$ and $\widehat{X^*}$ can be sampled without knowing y (and x^*).

- $\mathcal{B}_6$ simulates queries to H' as follows. When $\mathcal{A}$ queries H' for $\widehat{K}$, $\mathcal{B}_6$ computes $(\nu'', \mu'') \leftarrow \mathsf{PrivExt}_{\mathbb{H}}(\tau_{\mathbb{H}}, \widehat{K})$. Hence,

  $$\mathsf{Unwrap}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{K}) = (\nu'', \mu'') \cdot ([1]_G, [y]_G)^\intercal.$$

  To test whether $\widehat{K} = (\widehat{X^*})^y$ which in turn (implicitly) equals $\widehat{G}^{z \cdot y^2}$, $\mathcal{B}_6$ solves the equation

  $$z \cdot y^2 - \mu'' \cdot y - \nu'' = 0 \bmod p$$

for y. If one solution is the discrete logarithm of the given DL challenge game $\mathbf{G}_6$ aborts and $\mathcal{B}_6$ outputs y. (Note that due to $(\star)$, if the game does not abort, $\mathcal{A}$'s view is independent if it receives k_b or k_1.)

- $\mathcal{B}_6$ simulates queries to Dec as follows. As argued above, if $\mathsf{Dec}(\widehat{R}, \widehat{X}, s)$ does not return $\perp$, then $(\widehat{R}, \widehat{X}) \neq (\widehat{R^*}, \widehat{X^*})$. We have that

$$\mathsf{Unwrap}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{R}) = (\nu, \mu) \cdot ([1]_G, [y]_G)^\intercal \bmod p,$$
$$\mathsf{Unwrap}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{X}) = (\nu', \mu') \cdot ([1]_G, [y]_G)^\intercal \bmod p.$$

  If Dec does not return $\perp$, we have $s = r + c \cdot x \bmod p$ and hence

$$y \cdot (\mu + \mu' \cdot c) = s - \nu - \nu' \cdot c \bmod p. \tag{13.1}$$

  If $\mu + \mu' \cdot c \neq 0 \bmod p$, $\mathbf{G}_6$ aborts and $\mathcal{B}_6$ solves Equation (13.1) for y. If $\mu + \mu' \cdot c = 0 \bmod p$ and $\mu' \neq 0$ then both $\mathbf{G}_5$ and $\mathbf{G}_6$ abort. If $\mu + \mu' \cdot c = 0 \bmod p$ and $\mu' = 0$ then $\mu = 0$ and $dlog_{[1]_G}(\mathsf{Unwrap}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{X})) = x = \nu'$ allowing the reduction to simulate Dec response as $k := \widetilde{H}'(\widehat{Y}^x)$.

Therefore,

$$\left|\Pr\left[out_6 = 1\right] - \Pr\left[out_5 = 1\right]\right| \leqslant \mathrm{Adv}^{\mathsf{dl}}_{G,\mathcal{B}_6}(\lambda).$$

□

$\mathbf{G}_0$

$pp_G \leftarrow \mathsf{GGen}_G(1^\lambda)$
$(pp_{\mathbb{H}}, \tau_{\mathbb{H}}) \leftarrow \mathsf{GGen}_{\mathbb{H}}(pp_G, ([1]_G, [\beta_2]_G)^\intercal)$
$y \leftarrow \mathbb{Z}_p$
$\widehat{G} \leftarrow \mathsf{Rerand}_{\mathbb{H}}(pp_{\mathbb{H}}, \mathsf{Sam}_{\mathbb{H}}(pp_{\mathbb{H}}, 1))$
$\widehat{Y} \leftarrow \mathsf{Rerand}_{\mathbb{H}}(pp_{\mathbb{H}}, \mathsf{Sam}_{\mathbb{H}}(pp_{\mathbb{H}}, y))$
$pk \coloneqq (pp_{\mathbb{H}}, \widehat{G}, \widehat{Y})$
$T, T' \coloneqq []$
$x^*, r^* \leftarrow \mathbb{Z}_p$
$\widehat{X^*} \leftarrow \mathsf{Rerand}_{\mathbb{H}}(pp_{\mathbb{H}}, \mathsf{Sam}_{\mathbb{H}}(pp_{\mathbb{H}}, x^*))$
$\widehat{R^*} \leftarrow \mathsf{Rerand}_{\mathbb{H}}(pp_{\mathbb{H}}, \mathsf{Sam}_{\mathbb{H}}(pp_{\mathbb{H}}, r^*))$
$c^* \coloneqq \widetilde{H}(\widehat{R^*}, \widehat{X^*})$
$s^* \coloneqq r^* + c^* \cdot x^* \bmod p$
$k_0 \coloneqq \widetilde{H}'(\widehat{Y}^{x^*}), k_1 \leftarrow \mathcal{K}$
$b' \leftarrow \mathcal{A}^{H,H',\mathsf{Dec}}(pk, k_b, (\widehat{R^*}, \widehat{X^*}, s^*))$
return $b = b'$

(a) Description of $\mathbf{G}_0$.

$\mathsf{Dec}(\widehat{R}, \widehat{X}, s)$

if $\widehat{R} =_{\mathbb{H}} \widehat{R^*} \wedge \widehat{X} =_{\mathbb{H}} \widehat{X^*} \wedge s = s^*$ **then**
  return $\perp$
$c \coloneqq \widetilde{H}(\widehat{R}, \widehat{X})$
if $[s]_{\mathbb{H}} \neq_{\mathbb{H}} \widehat{R} \cdot \widehat{X}^c$ **then**
  return $\perp$
$k \coloneqq \widetilde{H}'(\widehat{X}^y)$
return k

(b) Decryption oracle associated with $\mathbf{G}_0$.

$H(\widehat{R}, \widehat{X})$

if $T[(\mathsf{GetID}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{R}), \mathsf{GetID}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{X}))] = \perp$ **then**
  $T[(\mathsf{GetID}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{R}), \mathsf{GetID}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{X}))] \leftarrow \mathbb{Z}_p$
return $T[(\mathsf{GetID}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{R}), \mathsf{GetID}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{X}))]$

(c) Random oracle H associated with $\mathbf{G}_0$.

$H'(\widehat{K})$

if $T'[\mathsf{GetID}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{K})] = \perp$ **then**
  $T'[\mathsf{GetID}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{K})] \leftarrow \mathcal{K}$
return $T'[\mathsf{GetID}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{K})]$

(d) Random oracle H' associated with $\mathbf{G}_0$.

Figure 13.2: The description of $\mathbf{G}_0$. $\mathbf{G}_0$ is identical to $\mathrm{Exp}^{\text{ind-cca2}}_{\mathsf{E}_{\mathsf{sElG}},\mathcal{A}}(\lambda)$.

$\mathbf{G}_1$

$pp_G \leftarrow \mathsf{GGen}_G(1^\lambda)$
$(pp_{\mathbb{H}}, \tau_{\mathbb{H}}) \leftarrow \mathsf{GGen}_{\mathbb{H}}(pp_G, ([1]_G, [\beta_2]_G)^\intercal)$
$y \leftarrow \mathbb{Z}_p$
$\widehat{G} \leftarrow \mathsf{Rerand}_{\mathbb{H}}(pp_{\mathbb{H}}, \mathsf{Sam}_{\mathbb{H}}(pp_{\mathbb{H}}, 1))$
$\widehat{Y} \leftarrow \mathsf{Rerand}_{\mathbb{H}}(pp_{\mathbb{H}}, \mathsf{Sam}_{\mathbb{H}}(pp_{\mathbb{H}}, y))$
$pk \coloneqq (pp_{\mathbb{H}}, \widehat{G}, \widehat{Y})$
$T, T' = []$
$z, r^* \leftarrow \mathbb{Z}_p, x^* \coloneqq z \cdot y$
$\widehat{X^*} \leftarrow \mathsf{Rerand}_{\mathbb{H}}(pp_{\mathbb{H}}, \mathsf{Sam}_{\mathbb{H}}(pp_{\mathbb{H}}, z \cdot y))$
$\widehat{R_1^*} \leftarrow \mathsf{Rerand}_{\mathbb{H}}(pp_{\mathbb{H}}, \mathsf{Sam}_{\mathbb{H}}(pp_{\mathbb{H}}, r^*))$
$c^* \coloneqq \widetilde{H}(\widehat{R^*}, \widehat{X^*})$
$s^* \coloneqq r^* + c^* \cdot z \cdot y \bmod p$
$\widehat{R_2^*} \leftarrow \mathsf{Rerand}_{\mathbb{H}}(pp_{\mathbb{H}},$
$\quad \mathsf{Sam}_{\mathbb{H}}(pp_{\mathbb{H}}, s^* - c^* \cdot x^*))$
$k_0 \coloneqq \widetilde{H}'(\widehat{Y}^{z \cdot y}), k_1 \leftarrow \mathcal{K}$
$b' \leftarrow \mathcal{A}^{H, H', \mathsf{Dec}}(pk, k_b, (\widehat{R_2^*}, \widehat{X^*}, s^*))$
return $b = b'$

(a) Description of $\mathbf{G}_1$.

$\mathsf{Dec}(\widehat{R}, \widehat{X}, s)$

if $\widehat{R} =_{\mathbb{H}} \widehat{R^*} \wedge \widehat{X} =_{\mathbb{H}} \widehat{X^*} \wedge s = s^*$ **then**
  return $\perp$
$c \coloneqq \widetilde{H}(\widehat{R}, \widehat{X})$
if $[s]_{\mathbb{H}} \neq_{\mathbb{H}} \widehat{R} \cdot \widehat{X}^c$ **then**
  return $\perp$
$k \coloneqq \widetilde{H}'(\widehat{X}^y)$
return k

(b) Decryption oracle associated with $\mathbf{G}_1$.

$H(\widehat{R}, \widehat{X})$

if $T[(\mathsf{GetID}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{R}), \mathsf{GetID}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{X}))] = \perp$ **then**
  $T[(\mathsf{GetID}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{R}), \mathsf{GetID}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{X}))] \leftarrow \mathbb{Z}_p$
return $T[(\mathsf{GetID}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{R}), \mathsf{GetID}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{X}))]$

(c) Random oracle H associated with $\mathbf{G}_1$.

$H'(\widehat{K})$

if $T'[\mathsf{GetID}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{K})] = \perp$ **then**
  $T'[\mathsf{GetID}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{K})] \leftarrow \mathcal{K}$
return $T'[\mathsf{GetID}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{K})]$

(d) Random oracle H' associated with $\mathbf{G}_1$.

Figure 13.3: The description of $\mathbf{G}_1$.

$\mathbf{G}_2$

$pp_G \leftarrow \mathsf{GGen}_G(1^\lambda)$
$(pp_{\mathbb{H}}, \tau_{\mathbb{H}}) \leftarrow \mathsf{GGen}_{\mathbb{H}}(pp_G, ([1]_G, [\beta_2]_G)^\intercal)$
$y \leftarrow \mathbb{Z}_p$
$\widehat{G} \leftarrow \mathsf{Rerand}_{\mathbb{H}}(pp_{\mathbb{H}}, \mathsf{Sam}_{\mathbb{H}}(pp_{\mathbb{H}}, 1))$
$\widehat{Y} \leftarrow \mathsf{Rerand}_{\mathbb{H}}(pp_{\mathbb{H}}, \mathsf{PrivSam}_{\mathbb{H}}(\tau_{\mathbb{H}}, y))$
$pk \coloneqq (pp_{\mathbb{H}}, \widehat{G}, \widehat{Y})$
$T, T' = []$
$z, r^* \leftarrow \mathbb{Z}_p, x^* \coloneqq z \cdot y$
$\widehat{X^*} \leftarrow \mathsf{Rerand}_{\mathbb{H}}(pp_{\mathbb{H}}, \mathsf{PrivSam}_{\mathbb{H}}(\tau_{\mathbb{H}}, z \cdot y))$
$\widehat{R_1^*} \leftarrow \mathsf{Rerand}_{\mathbb{H}}(pp_{\mathbb{H}}, \mathsf{Sam}_{\mathbb{H}}(pp_{\mathbb{H}}, r^*))$
$c^* \coloneqq \widetilde{H}(\widehat{R^*}, \widehat{X^*})$
$s^* \coloneqq r^* + c^* \cdot z \cdot y \bmod p$
$\widehat{R_2^*} \leftarrow \mathsf{Rerand}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{G}^{s^*} \cdot (\widehat{X^*})^{-c^*})$
$k_0 \coloneqq \widetilde{H}'(\widehat{Y}^{z \cdot y}), k_1 \leftarrow \mathcal{K}$
$b' \leftarrow \mathcal{A}^{H, H', \mathsf{Dec}}(pk, k_b, (\widehat{R_2^*}, \widehat{X^*}, s^*))$
return $b = b'$

(a) Description of $\mathbf{G}_2$.

$\mathsf{Dec}(\widehat{R}, \widehat{X}, s)$

if $\widehat{R} =_{\mathbb{H}} \widehat{R^*} \wedge \widehat{X} =_{\mathbb{H}} \widehat{X^*} \wedge s = s^*$ **then**
  return $\perp$
$c \coloneqq \widetilde{H}(\widehat{R}, \widehat{X})$
if $[s]_{\mathbb{H}} \neq_{\mathbb{H}} \widehat{R} \cdot \widehat{X}^c$ **then**
  return $\perp$
$k \coloneqq \widetilde{H}'(\widehat{X}^y)$
return k

(b) Decryption oracle associated with $\mathbf{G}_2$.

$H(\widehat{R}, \widehat{X})$

if $T[(\mathsf{GetID}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{R}), \mathsf{GetID}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{X}))] = \perp$ **then**
  $T[(\mathsf{GetID}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{R}), \mathsf{GetID}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{X}))] \leftarrow \mathbb{Z}_p$
return $T[(\mathsf{GetID}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{R}), \mathsf{GetID}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{X}))]$

(c) Random oracle H associated with $\mathbf{G}_2$.

$H'(\widehat{K})$

if $T'[\mathsf{GetID}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{K})] = \perp$ **then**
  $T'[\mathsf{GetID}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{K})] \leftarrow \mathcal{K}$
return $T'[\mathsf{GetID}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{K})]$

(d) Random oracle H' associated with $\mathbf{G}_2$.

Figure 13.4: The description of $\mathbf{G}_2$.

$\mathbf{G}_3$

$pp_G \leftarrow \mathsf{GGen}_G(1^\lambda)$
$y \leftarrow \mathbb{Z}_p$
$(pp_{\mathbb{H}}, \tau_{\mathbb{H}}) \leftarrow \mathsf{GGen}_{\mathbb{H}}(pp_G, ([1]_G, [y]_G)^\intercal)$
$\widehat{G} \leftarrow \mathsf{Rerand}_{\mathbb{H}}(pp_{\mathbb{H}}, \mathsf{Sam}_{\mathbb{H}}(pp_{\mathbb{H}}, 1))$
$\widehat{Y} \leftarrow \mathsf{Rerand}_{\mathbb{H}}(pp_{\mathbb{H}}, \mathsf{PrivSam}_{\mathbb{H}}(\tau_{\mathbb{H}}, (0,1)^\intercal))$
$pk \coloneqq (pp_{\mathbb{H}}, \widehat{G}, \widehat{Y})$
$T, T' = []$
$z, r^* \leftarrow \mathbb{Z}_p, x^* \coloneqq z \cdot y$
$\widehat{X^*} \leftarrow \mathsf{Rerand}_{\mathbb{H}}(pp_{\mathbb{H}}, \mathsf{PrivSam}_{\mathbb{H}}(\tau_{\mathbb{H}}, (0,z)^\intercal))$
$\widehat{R_1^*} \leftarrow \mathsf{Rerand}_{\mathbb{H}}(pp_{\mathbb{H}}, \mathsf{Sam}_{\mathbb{H}}(pp_{\mathbb{H}}, r^*))$
$c^* \coloneqq \widetilde{H}(\widehat{R^*}, \widehat{X^*})$
$s^* \coloneqq r^* + c^* \cdot z \cdot y \bmod p$
$\widehat{R_2^*} \leftarrow \mathsf{Rerand}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{G}^{s^*} \cdot (\widehat{X^*})^{-c^*})$
$k_0 \coloneqq \widetilde{H}'(\widehat{Y}^{z \cdot y}), k_1 \leftarrow \mathcal{K}$
$b' \leftarrow \mathcal{A}^{H, H', \mathsf{Dec}}(pk, k_b, (\widehat{R_2^*}, \widehat{X^*}, s^*))$
return $b = b'$

(a) Description of $\mathbf{G}_3$.

$\mathsf{Dec}(\widehat{R}, \widehat{X}, s)$

if $\widehat{R} =_{\mathbb{H}} \widehat{R^*} \wedge \widehat{X} =_{\mathbb{H}} \widehat{X^*} \wedge s = s^*$ **then**
  return $\perp$
$c \coloneqq \widetilde{H}(\widehat{R}, \widehat{X})$
if $[s]_{\mathbb{H}} \neq_{\mathbb{H}} \widehat{R} \cdot \widehat{X}^c$ **then**
  return $\perp$
$k \coloneqq \widetilde{H}'(\widehat{X}^y)$
return k

(b) Decryption oracle associated with $\mathbf{G}_3$.

$H(\widehat{R}, \widehat{X})$

if $T[(\mathsf{GetID}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{R}), \mathsf{GetID}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{X}))] = \perp$ **then**
  $T[(\mathsf{GetID}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{R}), \mathsf{GetID}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{X}))] \leftarrow \mathbb{Z}_p$
return $T[(\mathsf{GetID}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{R}), \mathsf{GetID}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{X}))]$

(c) Random oracle H associated with $\mathbf{G}_3$.

$H'(\widehat{K})$

if $T'[\mathsf{GetID}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{K})] = \perp$ **then**
  $T'[\mathsf{GetID}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{K})] \leftarrow \mathcal{K}$
return $T'[\mathsf{GetID}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{K})]$

(d) Random oracle H' associated with $\mathbf{G}_3$.

Figure 13.5: The description of $\mathbf{G}_3$.

$\mathbf{G}_4$

$pp_G \leftarrow \mathsf{GGen}_G(1^\lambda)$
$y \leftarrow \mathbb{Z}_p$
$(pp_{\mathbb{H}}, \tau_{\mathbb{H}}) \leftarrow \mathsf{GGen}_{\mathbb{H}}(pp_G, ([1]_G, [y]_G)^\intercal)$
$\widehat{G} \leftarrow \mathsf{Rerand}_{\mathbb{H}}(pp_{\mathbb{H}}, \mathsf{Sam}_{\mathbb{H}}(pp_{\mathbb{H}}, 1))$
$\widehat{Y} \leftarrow \mathsf{Rerand}_{\mathbb{H}}(pp_{\mathbb{H}}, \mathsf{PrivSam}_{\mathbb{H}}(\tau_{\mathbb{H}}, (0,1)^\intercal))$
$pk \coloneqq (pp_{\mathbb{H}}, \widehat{G}, \widehat{Y})$
$T, T' = [], U \coloneqq []$
$z, r^* \leftarrow \mathbb{Z}_p, x^* \coloneqq z \cdot y$
$\widehat{X^*} \leftarrow \mathsf{Rerand}_{\mathbb{H}}(pp_{\mathbb{H}}, \mathsf{PrivSam}_{\mathbb{H}}(\tau_{\mathbb{H}}, (0,z)^\intercal))$
$\widehat{R_1^*} \leftarrow \mathsf{Rerand}_{\mathbb{H}}(pp_{\mathbb{H}}, \mathsf{Sam}_{\mathbb{H}}(pp_{\mathbb{H}}, r^*))$
$c^* \coloneqq \widetilde{H}(\widehat{R^*}, \widehat{X^*})$
$s^* \coloneqq r^* + c^* \cdot z \cdot y \bmod p$
$\widehat{R_2^*} \leftarrow \mathsf{Rerand}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{G}^{s^*} \cdot (\widehat{X^*})^{-c^*})$
$k_0 \coloneqq \widetilde{H}'(\widehat{Y}^{z \cdot y}), k_1 \leftarrow \mathcal{K}$
$b' \leftarrow \mathcal{A}^{H, H', \mathsf{Dec}}(pk, k_b, (\widehat{R_2^*}, \widehat{X^*}, s^*))$
return $b = b'$

(a) Description of $\mathbf{G}_4$.

$\mathsf{Dec}(\widehat{R}, \widehat{X}, s)$

if $\widehat{R} =_{\mathbb{H}} \widehat{R_2^*} \wedge \widehat{X} =_{\mathbb{H}} \widehat{X^*} \wedge s = s^*$ **then**
  return $\perp$
$c \coloneqq \widetilde{H}(\widehat{R}, \widehat{X})$
if $[s]_{\mathbb{H}} \neq_{\mathbb{H}} \widehat{R} \cdot \widehat{X}^c$ **then**
  return $\perp$
$(\nu, \mu) \leftarrow \mathsf{PrivExt}_{\mathbb{H}}(\tau_{\mathbb{H}}, \widehat{R})$
$(\nu', \mu') \leftarrow \mathsf{PrivExt}_{\mathbb{H}}(\tau_{\mathbb{H}}, \widehat{X})$
if $U[(\mathsf{GetID}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{R}), \mathsf{GetID}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{X}))] \neq \perp$ **then**
  $(\nu, \mu, \nu', \mu') \coloneqq U[(\mathsf{GetID}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{R}), \mathsf{GetID}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{X}))]$
if $(\mu + \mu' \cdot c = 0) \wedge (\mu' \neq 0)$ **then**
  abort and output random bit
$k \coloneqq \widetilde{H}'(\widehat{X}^y)$
return k

(b) Decryption oracle associated with $\mathbf{G}_4$.

$H(\widehat{R}, \widehat{X})$

if $T[(\mathsf{GetID}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{R}), \mathsf{GetID}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{X}))] = \perp$ **then**
  $T[(\mathsf{GetID}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{R}), \mathsf{GetID}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{X}))] \leftarrow \mathbb{Z}_p$
  $U[(\mathsf{GetID}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{R}), \mathsf{GetID}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{X}))] \coloneqq (\mathsf{PrivExt}_{\mathbb{H}}(\tau_{\mathbb{H}}, \widehat{R}), \mathsf{PrivExt}_{\mathbb{H}}(\tau_{\mathbb{H}}, \widehat{X}))$
return $T[(\mathsf{GetID}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{R}), \mathsf{GetID}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{X}))]$

(c) Random oracle H associated with $\mathbf{G}_4$.

$H'(\widehat{K})$

if $T'[\mathsf{GetID}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{K})] = \perp$ **then**
  $T'[\mathsf{GetID}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{K})] \leftarrow \mathcal{K}$
return $T'[\mathsf{GetID}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{K})]$

(d) Random oracle H' associated with $\mathbf{G}_4$.

Figure 13.6: The description of $\mathbf{G}_4$. This game corresponds to G_1 from [FPS20].

$\mathbf{G}_5$

$pp_G \leftarrow \mathsf{GGen}_G(1^\lambda)$
$y \leftarrow \mathbb{Z}_p$
$(pp_{\mathbb{H}}, \tau_{\mathbb{H}}) \leftarrow \mathsf{GGen}_{\mathbb{H}}(pp_G, ([1]_G, [y]_G)^\intercal)$
$\widehat{G} \leftarrow \mathsf{Rerand}_{\mathbb{H}}(pp_{\mathbb{H}}, \mathsf{Sam}_{\mathbb{H}}(pp_{\mathbb{H}}, 1))$
$\widehat{Y} \leftarrow \mathsf{Rerand}_{\mathbb{H}}(pp_{\mathbb{H}}, \mathsf{PrivSam}_{\mathbb{H}}(\tau_{\mathbb{H}}, (0,1)^\intercal))$
$pk \coloneqq (pp_{\mathbb{H}}, \widehat{G}, \widehat{Y})$
$T, T' = [], U \coloneqq []$
$z, c^*, s^* \leftarrow \mathbb{Z}_p, x^* \coloneqq z \cdot y$
$\widehat{X^*} \leftarrow \mathsf{Rerand}_{\mathbb{H}}(pp_{\mathbb{H}}, \mathsf{PrivSam}_{\mathbb{H}}(\tau_{\mathbb{H}}, (0,z)^\intercal))$
$\widehat{R_2^*} \leftarrow \mathsf{Rerand}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{G}^{s^*} \cdot (\widehat{X^*})^{-c^*})$
if $T[(\mathsf{GetID}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{R_2^*}), \mathsf{GetID}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{X^*}))] = \bot$ **then**
  $T[(\mathsf{GetID}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{R_2^*}), \mathsf{GetID}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{X^*}))] \coloneqq c^*$
else
  abort and output random bit
$k_0 \coloneqq \widetilde{H}'(\widehat{Y}^{z \cdot y}), k_1 \leftarrow \mathcal{K}$
$b' \leftarrow \mathcal{A}^{H, H', \mathsf{Dec}}(pk, k_b, (\widehat{R_2^*}, \widehat{X^*}, s^*))$
return $b = b'$

(a) Description of $\mathbf{G}_5$.

$\mathsf{Dec}(\widehat{R}, \widehat{X}, s)$

if $\widehat{R} =_{\mathbb{H}} \widehat{R_2^*} \wedge \widehat{X} =_{\mathbb{H}} \widehat{X^*} \wedge s = s^*$ **then**
  return $\bot$
$c \coloneqq \widetilde{H}(\widehat{R}, \widehat{X})$
if $[s]_{\mathbb{H}} \neq_{\mathbb{H}} \widehat{R} \cdot \widehat{X}^c$ **then**
  return $\bot$
$(\nu, \mu) \leftarrow \mathsf{PrivExt}_{\mathbb{H}}(\tau_{\mathbb{H}}, \widehat{R})$
$(\nu', \mu') \leftarrow \mathsf{PrivExt}_{\mathbb{H}}(\tau_{\mathbb{H}}, \widehat{X})$
if $U[(\mathsf{GetID}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{R}), \mathsf{GetID}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{X}))] \neq \bot$ **then**
  $(\nu, \mu, \nu', \mu') \coloneqq U[(\mathsf{GetID}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{R}), \mathsf{GetID}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{X}))]$
if $(\mu + \mu' \cdot c = 0) \wedge (\mu' \neq 0)$ **then**
  abort and output random bit
$k \coloneqq \widetilde{H}'(\widehat{X}^y)$
return k

(b) Decryption oracle associated with $\mathbf{G}_5$.

$H(\widehat{R}, \widehat{X})$

if $T[(\mathsf{GetID}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{R}), \mathsf{GetID}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{X}))] = \bot$ **then**
  $T[(\mathsf{GetID}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{R}), \mathsf{GetID}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{X}))] \leftarrow \mathbb{Z}_p$
  $U[(\mathsf{GetID}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{R}), \mathsf{GetID}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{X}))] \coloneqq$
    $(\mathsf{PrivExt}_{\mathbb{H}}(\tau_{\mathbb{H}}, \widehat{R}), \mathsf{PrivExt}_{\mathbb{H}}(\tau_{\mathbb{H}}, \widehat{X}))$
return $T[(\mathsf{GetID}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{R}), \mathsf{GetID}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{X}))]$

(c) Random oracle H associated with $\mathbf{G}_5$.

$H'(\widehat{K})$

if $T'[\mathsf{GetID}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{K})] = \bot$ **then**
  $T'[\mathsf{GetID}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{K})] \leftarrow \mathcal{K}$
return $T'[\mathsf{GetID}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{K})]$

(d) Random oracle H' associated with $\mathbf{G}_5$.

Figure 13.7: The description of $\mathbf{G}_5$. This game corresponds to G_2 from [FPS20].

$\mathbf{G}_6$

$pp_G \leftarrow \mathsf{GGen}_G(1^\lambda)$
$y \leftarrow \mathbb{Z}_p$
$(pp_{\mathbb{H}}, \tau_{\mathbb{H}}) \leftarrow \mathsf{GGen}_{\mathbb{H}}(pp_G, ([1]_G, [y]_G)^\intercal)$
$\widehat{G} \leftarrow \mathsf{Rerand}_{\mathbb{H}}(pp_{\mathbb{H}}, \mathsf{Sam}_{\mathbb{H}}(pp_{\mathbb{H}}, 1))$
$\widehat{Y} \leftarrow \mathsf{Rerand}_{\mathbb{H}}(pp_{\mathbb{H}}, \mathsf{PrivSam}_{\mathbb{H}}(\tau_{\mathbb{H}}, (0,1)^\intercal))$
$pk \coloneqq (pp_{\mathbb{H}}, \widehat{G}, \widehat{Y})$
$T, T' = [], U \coloneqq []$
$z, c^*, s^* \leftarrow \mathbb{Z}_p, x^* \coloneqq z \cdot y$
$\widehat{X^*} \leftarrow \mathsf{Rerand}_{\mathbb{H}}(pp_{\mathbb{H}}, \mathsf{PrivSam}_{\mathbb{H}}(\tau_{\mathbb{H}}, (0,z)^\intercal))$
$\widehat{R_2^*} \leftarrow \mathsf{Rerand}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{G}^{s^*} \cdot (\widehat{X^*})^{-c^*})$
if $T[(\mathsf{GetID}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{R_2^*}),$
$\quad \mathsf{GetID}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{X^*}))] = \perp$ **then**
$\quad T[(\mathsf{GetID}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{R_2^*}),$
$\quad \mathsf{GetID}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{X^*}))] \coloneqq c^*$
else
$\quad$ **abort and output random bit**
$k_0 \coloneqq \widetilde{H}'(\widehat{Y}^{z \cdot y}), k_1 \leftarrow \mathcal{K}$
$b' \leftarrow \mathcal{A}^{H, H', \mathsf{Dec}}(pk, k_1, (\widehat{R_2^*}, \widehat{X^*}, s^*))$
return $b = b'$

(a) Description of $\mathbf{G}_6$.

$\mathsf{Dec}(\widehat{R}, \widehat{X}, s)$

if $\widehat{R} =_{\mathbb{H}} \widehat{R_2^*} \wedge \widehat{X} =_{\mathbb{H}} \widehat{X^*} \wedge s = s^*$ **then**
$\quad$ **return** $\perp$
$c \coloneqq \widetilde{H}(\widehat{R}, \widehat{X})$
if $[s]_{\mathbb{H}} \neq_{\mathbb{H}} \widehat{R} \cdot \widehat{X}^c$ **then**
$\quad$ **return** $\perp$
$(\nu, \mu) \leftarrow \mathsf{PrivExt}_{\mathbb{H}}(\tau_{\mathbb{H}}, \widehat{R})$
$(\nu', \mu') \leftarrow \mathsf{PrivExt}_{\mathbb{H}}(\tau_{\mathbb{H}}, \widehat{X})$
if $\mu + \mu' \cdot c \neq 0$ **then**
$\quad$ **abort and output random bit** $(\star\star)$
if $U[(\mathsf{GetID}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{R}),$
$\quad \mathsf{GetID}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{X}))] \neq \perp$ **then**
$\quad (\nu, \mu, \nu', \mu') \coloneqq U[(\mathsf{GetID}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{R}),$
$\quad\quad \mathsf{GetID}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{X}))]$
if $(\mu + \mu' \cdot c = 0) \wedge (\mu' \neq 0)$ **then**
$\quad$ **abort and output random bit**
$k \coloneqq \widetilde{H}'(\widehat{X}^y)$
return k

(b) Decryption oracle associated with $\mathbf{G}_6$.

$H(\widehat{R}, \widehat{X})$

if $T[(\mathsf{GetID}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{R}), \mathsf{GetID}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{X}))] = \perp$ **then**
$\quad T[(\mathsf{GetID}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{R}), \mathsf{GetID}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{X}))] \leftarrow \mathbb{Z}_p$
$\quad U[(\mathsf{GetID}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{R}), \mathsf{GetID}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{X}))] \coloneqq$
$\quad\quad (\mathsf{PrivExt}_{\mathbb{H}}(\tau_{\mathbb{H}}, \widehat{R}), \mathsf{PrivExt}_{\mathbb{H}}(\tau_{\mathbb{H}}, \widehat{X}))$
return $T[(\mathsf{GetID}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{R}), \mathsf{GetID}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{X}))]$

(c) Random oracle H associated with $\mathbf{G}_6$.

$H'(\widehat{K})$

if $\widehat{K} = \widehat{X^*}^y$ **then**
$\quad$ **abort and output random bit** $(\star)$
if $T'[\mathsf{GetID}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{K})] = \perp$ **then**
$\quad T'[\mathsf{GetID}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{K})] \leftarrow \mathcal{K}$
return $T'[\mathsf{GetID}_{\mathbb{H}}(pp_{\mathbb{H}}, \widehat{K})]$

(d) Random oracle H' associated with $\mathbf{G}_6$.

Figure 13.8: The description of $\mathbf{G}_6$. This game corresponds to G_3 from [FPS20].

Part III

ON PSEUDORANDOM ENCODINGS

14 INTRODUCTION

In this part, we present the results of [ACI+20]. This part is taken verbatim from [ACI+20] with minor changes.

The problem of *compression* has been extensively studied in the field of information theory and, more recently, in computational complexity and cryptography [GS85; Wee04; TVZ05; HLR07]. Informally, given a distribution X, compression aims to efficiently encode samples from X as short strings while at the same time being able to efficiently recover these samples. While the typical information-theoretic study of compression considers the case of compressing multiple independent samples from the same source X, its study in computer science, and in particular in this part, considers the "single-shot" case. Compression in this setting is closely related to *randomness condensers* [RR99; TV00; TUZ01; DRV12] and *resource-bounded Kolmogorov complexity* [Sol64; Kol68; Cha69; LV90; LV19] – two well-studied problems in computational complexity. Randomness condensers, which relax randomness extractors, are functions that efficiently map an input distribution into an output distribution with a higher entropy rate. A randomness condenser can be viewed as an efficient compression algorithm, without a corresponding efficient decompression algorithm. The resource-bounded Kolmogorov complexity of a string is the smallest description length of an efficient program that outputs this string. This program description can be viewed as a compressed string, such that decoding is efficiently possible, while finding the compressed string may be inefficient.

An important property of efficient compression algorithms, which combines the efficiency features of randomness condensers and resource-bounded Kolmogorov complexity, is their ability to efficiently produce "random-looking" outputs while allowing the original input to be efficiently recovered. Despite the large body of work on compression and its computational variants, this fundamental property has, to our knowledge, never been the subject of a dedicated study. In this part, we fill this gap by initiating such a study. Before formalizing the problem, we give a simple motivating example.

Consider the goal of encrypting a sample x from a distribution X (say, a random 5-letter English word from the Merriam-Webster Dictionary) using a low-entropy secret key k. Applying a standard symmetric-key encryption scheme with a key derived from k gives rise to the following brute-force attack: Try to decrypt with different keys until obtaining x' in the support of X. In the typical case that wrong keys always lead to x' outside the support of X, this attack successfully recovers x. Variants of this attack arise in different scenarios, including password-authenticated key exchange [BM92], honey encryption [JR14], subliminal communication and steganography [vHL05; CGOS07; HPRV19], and more. A natural solution is to use perfect compression: if x can be compressed to a *uniformly random* string $\hat{x} \in \{0,1\}^n$ before being encrypted, it cannot be distinguished from another random string $\hat{x}' \in \{0,1\}^n$ obtained by trying the wrong key. Note, however, that compression may be an overkill for this application. Instead, it suffices to efficiently

encode x into a (possibly longer) *pseudorandom* string from which x can be efficiently decoded. This more general solution motivates the question we consider in this part.

ENCODING INTO THE UNIFORM DISTRIBUTION. We initiate the study of encoding distributions into a random-looking distribution. Informally, we say that a distribution ensemble X_λ admits a *pseudorandom encoding* if there exist efficient encoding and decoding algorithms (E_X, D_X), where D_X is deterministic, such that

$$\Pr\left[y \leftarrow X_\lambda : D_X(E_X(y)) = y\right] \text{ is overwhelming and} \tag{14.1}$$

$$\left\{y \leftarrow X_\lambda : E_X(y)\right\} \approx U_{n(\lambda)}. \tag{14.2}$$

Here, "$\approx$" denotes some notion of indistinguishability (we will consider both computational and statistical indistinguishability), and the probability is over the randomness of both E_X and X_λ. The polynomial $n(\lambda)$ denotes the output length of the encoding algorithm E_X. We refer to Equation (14.1) as *correctness* and to Equation (14.2) as *pseudorandomness*. It will also be useful to consider distribution ensembles parameterized by an input m from a language L. We say that such a distribution ensemble $(X_m)_{m\in L}$ admits a pseudorandom encoding if there exist efficient algorithms (E_X, D_X) as above satisfying correctness and pseudorandomness for all $m \in L$, where E_X and D_X both additionally receive m as input. Note that we insist on the decoding algorithm being efficient. This is required for our motivating applications.[53] Note also that encoding and decoding above are *keyless*; that is, we want encoded samples to be close to uniform *even though anyone can decode them*. This is a crucial distinction from, for instance, encryption schemes with pseudorandom ciphertexts, which look uniformly distributed to everyone except the owner of the decryption key, and cannot be efficiently decrypted without knowing the decryption key. Here, we seek to simultaneously achieve pseudorandomness and correctness for all parties.

[53] *Without this requirement, the problem can be solved using non-interactive commitment schemes with the additional property that commitments are pseudorandom (which exist under standard cryptographic assumptions).*

Our motivation for studying pseudorandom encodings stems from the fact that this very natural problem appears in a wide variety of – sometimes seemingly unrelated – problems in cryptography. We already mentioned steganography, honey encryption, and password-authenticated key exchange; we will cover more such connections in this part. Yet, this notion of encoding has to our knowledge never been studied systematically. In this part, we study several natural flavors of this notion, obtain positive and negative results about realizing them, and map their connections with other problems in cryptography.

The main focus of this part is on the hypothesis that *all* efficiently samplable distributions admit a pseudorandom encoding. Henceforth, we refer to this hypothesis as the *pseudorandom encoding hypothesis* (PREH).

For describing our results, it will be convenient to use the following general notion of efficiently samplable distributions. A distribution family ensemble $(X_m)_{m\in L}$ (for some language $L \subseteq \{0,1\}^*$) is efficiently samplable if there exists a probabilistic polynomial time (PPT) algorithm S such that $S(m)$ is distributed according to X_m for every $m \in L$. In case the distribution does not depend on additional inputs, L can be considered equal to $\mathbb{N}$.

OVERVIEW OF CONTRIBUTIONS. Following is a brief summary of our main contributions along with pointers to the relevant technical chapters. We

will give an expanded overview of the contributions and the underlying techniques in the rest of this chapter.

- We provide a unified study of different flavors of pseudorandom encodings (PRE) and identify computational, randomized PRE in the CRS model as a useful and achievable notion (Chapter 17 and Theorem 17.13).
- We establish a two-way relation between PRE and the previously studied notion of invertible sampling (Theorem 16.1). This reveals unexpected connections between seemingly unrelated problems in cryptography (e. g., between adaptively secure computation for general functionalities and "honey encryption").
- We bootstrap "adaptive PRE" from "static PRE" (Theorem 15.1). As a consequence, one can base succinct adaptively secure computation on standard IO as opposed to subexponential IO [CsW19] (Corollary 16.3).
- In [ACI+20], we use PRE to obtain a compiler from standard secure multiparty computation (MPC) protocols to *covert* MPC protocols. This result is not part of this thesis.

14.1 FLAVORS OF PSEUDORANDOM ENCODINGS

The notion of pseudorandom encoding has several natural flavors, depending on whether the encoding algorithm is allowed to use randomness or not, and whether the pseudorandomness property satisfies a computational or information-theoretic notion of indistinguishability. We denote the corresponding hypotheses that every efficiently samplable distribution can be pseudorandomly encoded according to the above variants as $\mathsf{PREH}^{\mathsf{rand}}_{\approx_c}$, $\mathsf{PREH}^{\mathsf{rand}}_{\equiv_s}$, $\mathsf{PREH}^{\mathsf{det}}_{\approx_c}$ and $\mathsf{PREH}^{\mathsf{det}}_{\equiv_s}$.[54]

We note that not all efficiently samplable distributions can be pseudorandomly encoded with a deterministic encoding algorithm. For instance, a distribution which has one very likely event and many less likely ones requires one specific encoding to appear with high probability. Thus, we formally restrict the deterministic variants of the pseudorandom encoding hypothesis to only hold for "compatible" samplers, which still results in interesting connections. In this overview, however, we ignore this restriction.

Further, we explore relaxations which rely on a trusted setup assumption: we consider the pseudorandom encoding hypothesis in the *common reference string model*, in which a common string sampled in a trusted way from some distribution is made available to the parties. This is the most common setup assumption in cryptography and it is standard to consider the feasibility of cryptographic primitives in this model to overcome limitations in the plain model. That is, we ask whether for every efficiently samplable distribution X, there exists an efficiently samplable CRS distribution and efficient encoding and decoding algorithms $(\mathsf{E}_X, \mathsf{D}_X)$ as above, such that correctness and pseudorandomness hold, where the encoding and decoding algorithm as well as the distinguisher receive the CRS as input, and the distributions in Equations (14.1) and (14.2) are additionally over the choice of the CRS.

Considering distributions which may depend on an input $m \in L$ further entails two different flavors. On the one hand, we consider the notion where

[54] *The subscripts $\equiv_s$ and $\approx_c$ denote statistical and computational guarantees, respectively. The superscripts* det *and* rand *indicate whether the encoding algorithm is deterministic or randomized.*

inputs m are chosen adversarially but *statically* (that is, independent of the CRS) and, on the other hand, we consider the stronger notion where inputs m are chosen adversarially and *adaptively* depending on the CRS. We henceforth denote these variants by the prefix "c" and "ac", respectively.[55]

[55] *The prefix "c" denotes the static variants with a CRS and the prefix "ac" denotes the adaptive variants with a CRS.*

STATIC-TO-ADAPTIVE TRANSFORMATION. The adaptive notion, where inputs may be chosen depending on the CRS, is clearly stronger than the static notion. However, surprisingly, the very nature of pseudorandom encodings allows one to apply an indirection argument similar to the one used in [HJK+16; CPR17; CPV17], which yields a static-to-adaptive transformation.

THEOREM (informal). *If all efficiently samplable distributions can be pseudorandomly encoded in the CRS model with a* static *choice of inputs, then all efficiently samplable distributions can be pseudorandomly encoded in the CRS model with an* adaptive *choice of inputs.*

Static-to-adaptive transformations in cryptography are generally nontrivial, and often come at a big cost in security when they rely on a "complexity leveraging" technique. This connection and its application we will discuss below are a good demonstration of the usefulness of the notion of pseudorandom encodings.

RELAXING COMPRESSION. The notion of statistical deterministic pseudorandom encodings recovers the notion of perfect compression. Hence, this conflicts with the existence of one-way functions.[56] In our systematic study of pseudorandom encodings, we gradually relax perfect compression in several dimensions, while maintaining one crucial property – the indistinguishability of the encoded distribution from true randomness.

[56] *If perfect compression exists, pseudorandom generators cannot exist (observation attributed to Levin in [GS85]).*

EXAMPLE. To illustrate the importance of this property, we elaborate on the example we outline at the beginning of this chapter, focusing more specifically on password-authenticated key exchange (PAKE). A PAKE protocol allows two parties holding a (low entropy) common password to jointly and confidentially generate a (high entropy) secret key, such that the protocol is resilient against offline dictionary attacks, and no adversary can establish a shared key with a party if he does not know the matching password. A widely used PAKE protocol due to Bellovin and Merritt [BM92] has a very simple structure: the parties use their low-entropy password to encrypt the flows of a key-exchange protocol using a block cipher. When the block cipher is modeled as a random cipher, it has the property that decrypting a ciphertext (of an arbitrary plaintext) under an incorrect secret key yields a fresh random plaintext. Thus, Bellovin and Merritt point out that the security of their PAKE protocol requires that "the message to be encrypted by the password must be indistinguishable from a random number." This is easy to achieve for Diffie-Hellman key exchange over the multiplicative group of integers modulo a prime p. However, for elliptic curve groups this is no longer the case, and one needs to resort to alternative techniques including nontrivial *point compression* algorithms that compress the representation of a random group element into a nearly uniform bitstring [BMN01].

Clearly, our relaxation of compression does not preserve the useful property of obtaining outputs that are *shorter* than the inputs. However, the

remaining pseudorandomness property is good enough for many applications.

In the following, we elaborate on our weakest notion of pseudorandom encodings, that is, pseudorandom encodings allowing the encoding algorithm to be randomized and providing a computational pseudorandomness guarantee. We defer the discussion on the stronger statistical or deterministic variants to Section 14.3, where we derive negative results for most of these stronger notions, which leaves computational randomized pseudorandom encodings as the "best possible" notion that can be realized for general distributions.

14.1.1 Randomized, Computational Pseudorandom Encodings

Computational randomized pseudorandom encodings allow the encoding algorithm to be randomized and require only computational pseudorandomness.

RELATION TO INVERTIBLE SAMPLING. We show a simple but unexpected connection with the notion of *invertible sampling* [CFGN96; DN00; GKM+00]. Informally, invertible sampling refers to the task of finding, given samples from a distribution, random coins that *explain* the sample. Invertible sampling allows to *obliviously* sample from distributions, that is, sampling from distributions *without knowing the corresponding secrets*. This can be useful for, e. g., sampling common reference strings without knowing the random coins or public keys without knowing the corresponding secret keys. A natural relaxation of this notion was systematically studied by Ishai, Kumarasubramanian, Orlandi, and Sahai [IKOS10]. Concretely, a PPT sampler S is *inverse samplable* if there exists an alternative PPT sampler $\overline{S}$ and a PPT inverse sampler $\overline{S}^{-1}$ such that

$$\{y \leftarrow S(1^\lambda) : y\} \approx_c \{y \leftarrow \overline{S}(1^\lambda) : y\},$$
$$\{y \leftarrow \overline{S}(1^\lambda; r) : (r, y)\} \approx_c \{y \leftarrow \overline{S}(1^\lambda) : (\overline{S}^{-1}(1^\lambda, y), y)\}.$$

Note that the inverse sampling algorithm is only required to efficiently inverse-sample from another distribution $\overline{S}$, but this distribution must be computationally close to the distribution induced by S. The main question studied in [IKOS10] is whether *every* efficient sampler admits such an invertible sampler. They refer to this hypothesis as the *invertible sampling hypothesis* (ISH), and show that ISH is equivalent to adaptive MPC for general randomized functionalities that may hide their internal randomness. In this part, we show the following two-way relation with pseudorandom encoding.

THEOREM (informal). *A distribution admits a pseudorandom encoding* if and only if *it admits invertible sampling.*

Intuitively, the efficient encoding algorithm corresponds to the inverse sampling algorithm, and decoding an encoded string corresponds to sampling with the de-randomized alternative sampler $\overline{S}$. This equivalence immediately extends to all variants of pseudorandom encodings and corresponding variants of invertible sampling we introduce in this part. Invertible sampling is itself connected to other useful cryptographic notions, such as oblivious sampling, trusted common reference string generations, and adaptively secure computation (which we will elaborate upon below).

Building on this connection, the impossibility result of [IKOS10] translates to our setting. On a high level, extractable one-way functions (EOWFs) conflict with invertible sampling because they allow to extract a "secret" (in this case a pre-image) from an image, independently of how it was computed. This conflicts with invertible sampling because invertible sampling is about sampling without knowing the secrets.

THEOREM (informal, [IKOS10]). *Assuming the existence of extractable one-way functions (EOWF) and a non-interactive zero-knowledge proof system,* $\mathsf{PREH}^{\mathsf{rand}}_{\approx_c}$ *does not hold.*

This suggests that towards a realizable notion of pseudorandom encodings, a further relaxation is due. Thus, we ask whether the above impossibility result extends to the CRS model. In the CRS model, the above intuition why ISH conflicts with EOWFs fails, because the CRS can contain an obfuscated program that samples an image using some secret, but does not output this secret.

Dachman-Soled, Katz, and Rao [DKR15] (building on the universal deniable encryption construction of Sahai and Waters [SW14]) construct a so-called "explainability compiler" that implies $\mathsf{cISH}^{\mathsf{rand}}_{\approx_c}$ based on indistinguishability obfuscation (IO). By our equivalence theorem above, this implies pseudorandom encodings for all efficiently samplable distributions in the CRS model, with static choice of inputs, from IO. Invoking the static-to-adaptive transformation detailed above, this also applies to the adaptive variant.

THEOREM (informal). *Assuming the existence of (polynomially secure) indistinguishability obfuscation and one-way functions,* $\mathsf{acPREH}^{\mathsf{rand}}_{\approx_c}$ *holds.*

Note that [IKOS10] claim that their impossibility result extends to the CRS model, whereas the above theorem seems to suggest the opposite. We show that technically the result of [IKOS10] does extend to the CRS model at the cost of assuming *unbounded auxiliary-input* extractable one-way functions, a strong flavor of EOWFs that seems very unlikely to exist but cannot be unconditionally ruled out.

THEOREM (informal). *Assuming the existence of extractable one-way functions with* unbounded *common auxiliary input and a non-interactive zero-knowledge proof system,* $\mathsf{cPREH}^{\mathsf{rand}}_{\approx_c}$ *does not hold.*

In fact, this apparent contradiction has been the source of some confusion in previous works: the work of [IKOS10] makes an informal claim that their impossibility result for ISH extends to the CRS model. However, due to the connection between ISH and adaptively secure MPC (which we will discuss in more details later on), this claim was challenged in [DKR15]: the authors achieve a construction of adaptively secure MPC for all functionalities assuming IO, which seemingly contradicts the claim of [IKOS10]. The authors of [DKR15] therefore stated that the "impossibility result of Ishai et al. [...] does not hold in the CRS model." Our extension clarifies that the distinction is in fact more subtle: the result of [IKOS10] *does* extend to the CRS model, but at the cost of assuming EOWF with *unbounded auxiliary inputs*. This does not contradict the constructions based on IO, because IO and EOWF with unbounded auxiliary inputs are known to be contradictory [BCPR16].

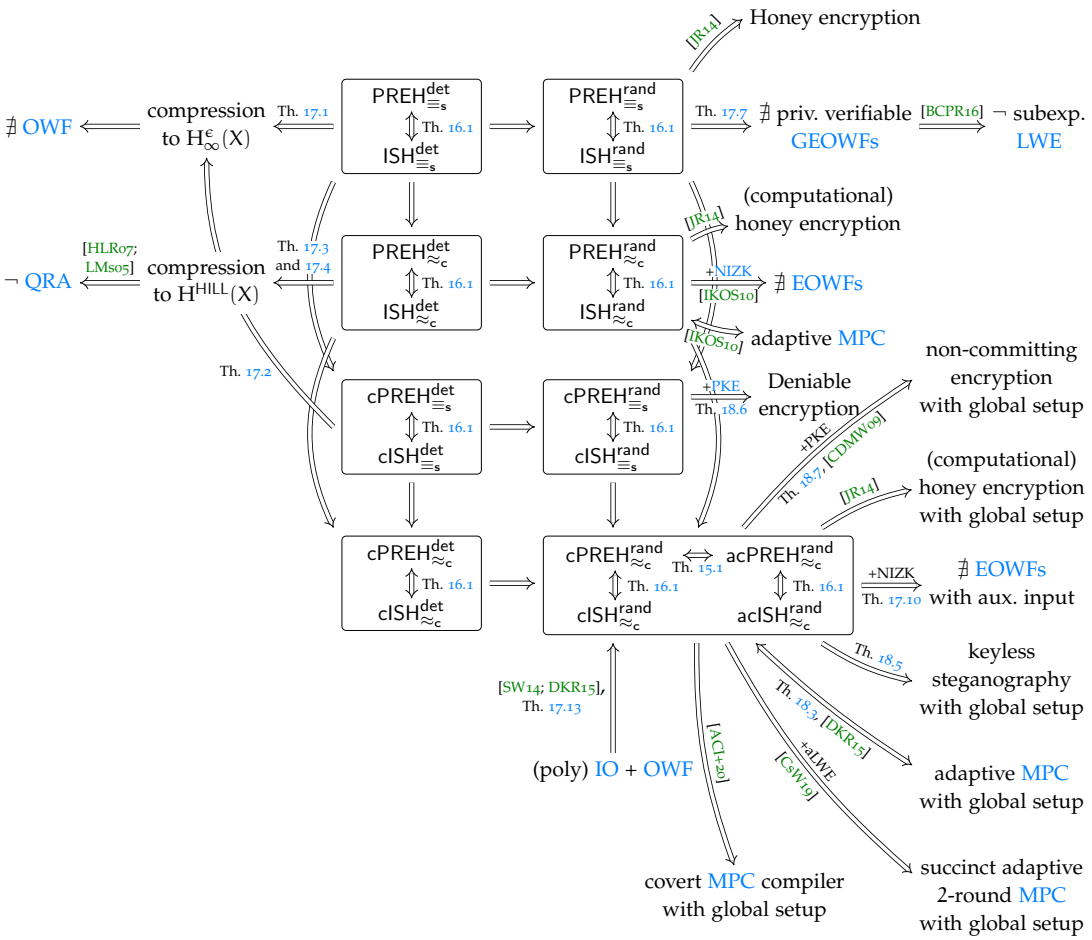


Figure 14.1: An overview of the connections of the pseudorandom encoding hypothesis and other fields of cryptography and computational complexity theory. For simplicity, our static-to-adaptive transformation only appears in the computational, randomized setting in this overview, but also applies to the other settings. (Since the deterministic variants of the pseudorandom encoding hypothesis are impossible for some pathologic samplers, the arrows between deterministic and randomized variants of the pseudorandom encoding hypothesis are to be read as if the deterministic variant is true *for some sampler*, then the corresponding randomized variant is true *for that sampler*.)

OVERVIEW. In Figure 14.1, we provide a general summary of the many flavors of the pseudorandom encoding hypothesis, and how they relate to a wide variety of other primitives.

FURTHER RELAXATION. We further study an additional relaxation of pseudorandom encodings, where we allow the encoding algorithm to run in super-polynomial time. We show that this relaxed variant can be achieved from cryptographic primitives similar to *extremely lossy functions* [Zha16], which can be based on the exponential hardness of the decisional Diffie-Hellman problem – a strong assumption, but (still) more standard than indistinguishability obfuscation. However, the applicability of the resulting notion turns out to be rather restricted.

14.2 IMPLICATIONS OF OUR RESULTS

In this section, we elaborate on the implications of the techniques we develop and the results we obtain for a variety of other cryptographic primitives.

14.2.1 New Results for Adaptively Secure Computation

As mentioned above, a sampler admits invertible sampling if and only if it can be pseudorandomly encoded. Ishai et al. [IKOS10] establish a two-way connection between invertible sampling and *adaptively* secure MPC for general randomized functionalities. An MPC protocol allows two or more parties to jointly evaluate a (possibly randomized) functionality $\mathcal{F}$ on their inputs without revealing anything to each other except what follows from their inputs and outputs. This should hold even in the presence of an adversary who can corrupt *any* number of parties in an adaptive (sequential) fashion. When we write "adaptive MPC", we mean adaptive MPC for *all* randomized functionalities. This should be contrasted with weaker notions of adaptive MPC for strict subsets of corrupted parties [BH93; CFGN96; GS12] or for adaptively well-formed functionalities[57] [CLOS02] which can both be achieved from mild assumptions. The connection from [IKOS10] shows that adaptive MPC for all randomized functions is possible if and only if every PPT sampler admits invertible sampling, i. e., the invertible sampling hypothesis is true.

[57] *Adaptively well-formed functionalities do not hide internal randomness.*

We show that this result generalizes to the global CRS model. More precisely, we prove the adaptive variant of the pseudorandom encoding hypothesis in the CRS model $\mathsf{acPREH}^{\mathsf{rand}}_{\approx_c}$ to be equivalent to adaptive MPC in the global CRS model.[58]

[58] *Together with the conflict between $\mathsf{cPREH}^{\mathsf{rand}}_{\approx_c}$ and EOWFs with unbounded auxiliary input, this corrects a claim made in [DKR15] that the impossibility result of adaptive MPC from [IKOS10] would not extend to the CRS model.*

As detailed above, the static variant of the pseudorandom encoding hypothesis $\mathsf{cPREH}^{\mathsf{rand}}_{\approx_c}$ in the CRS model follows from IO (and one-way functions). Applying our static-to-adaptive transformation, the same holds for the adaptive variant. Thus, we obtain the first instantiation of an adaptive explainability compiler [DKR15] without complexity leveraging and, hence, based only on polynomial hardness assumptions. The recent work of Cohen, shelat, and Wichs [CsW19] uses such an adaptive explainability compiler to obtain succinct adaptive MPC, where "succinct" means that the communication complexity is sublinear in the complexity of the evaluated function. Due to our instantiation of $\mathsf{acPREH}^{\mathsf{rand}}_{\approx_c}$ from polynomial IO, we improve the results of [CsW19] by relaxing the requirement for subexponentially secure IO to polynomially secure IO in a black-box way.

COROLLARY (informal). *Assuming the existence of polynomially secure indistinguishability obfuscation and the adaptive hardness of the learning with errors problem, then malicious, two-round, UC-secure adaptive MPC and sublinear communication complexity is possible (in the local CRS model, for all deterministic functionalities).*

14.2.2 Steganography and Covert Multi-Party Computation

We explore the connection of the pseudorandom encoding hypothesis to various flavors of steganography. The goal of steganography, informally, is to embed secret messages in distributions of natural-looking messages, in

order to hide them from external observers. While the standard setting for steganography relies on shared secret keys to encode the messages, we show that pseudorandom encodings naturally give rise to a strong form of *keyless steganography*. Namely, one can rely on pseudorandom encodings to encode any message into an innocent-looking distribution, without truly hiding the message (since anyone can decode the stream), but providing *plausible deniability*, in the sense that, even with the decoded message, it is impossible to tell apart whether this message was indeed encoded by the sender, or whether it is simply the result of decoding the innocent distribution.

COROLLARY (informal). *Assuming pseudorandom encodings, then there exists a keyless steganographic protocol which provides plausible deniability.*

Plausible deniability is an important security notion; in particular, an important cryptographic primitive in this area is the notion of (sender-)deniable encryption [CDNO97], which is known to exist assuming indistinguishability obfuscation [SW14]. Deniable encryption enables to "explain" ciphertexts produced for some message to any arbitrary other message by providing corresponding random coins for a faked encryption process. We view it as an interesting open problem to build deniable encryption under the pseudorandom encoding hypothesis together with more standard cryptographic primitives; we make a first step in this direction and show the following: the *statistical* variant of pseudorandom encodings, together with the existence of public-key encryption, implies deniable encryption. Interestingly, we also show that the computational randomized pseudorandom encoding hypothesis suffices to imply non-committing encryption, a weaker form of deniable encryption allowing to explain only *simulated* ciphertexts to arbitrary messages [CFGN96].

COVERT SECURE COMPUTATION. Covert MPC [vHL05; CGOS07] is an intriguing flavor of MPC that aims at achieving the following strong security guarantee: if the output of the protocol is not "favorable", the transcript of the interaction should not leak any information to the parties, including whether any given party was *actually taking part in the protocol*. This strong form of MPC aims at providing security guarantees when the very act of starting a computation with other parties should remain hidden. As an example, [vHL05] suppose that a CIA agent who infiltrated a terrorist group wants to make a handshake with another individual to find out whether she is also a CIA agent. Here, we show that pseudorandom encodings give rise to a general compiler transforming a standard MPC protocol into a covert one, in a round-preserving way. The idea is to encode each round of the protocol such that encoded messages look random. Together with the equivalence between adaptively secure MPC and pseudorandom encodings, this gives a connection between two seemingly unrelated notions of secure computation.

COROLLARY (informal). *Assuming adaptively secure MPC for all functionalities, there exists a round-preserving compiler that transforms a large class of "natural" MPC protocols into covert MPC protocols (in the static, semi-honest setting).*

We emphasize that the result on covert secure computation is *not* part of this thesis. We refer the reader to the original publication [ACI+20] for more details on this result.

14.2.3 Other Results

Due to our infeasibility results of $\mathsf{PREH}^{\mathsf{rand}}_{\equiv_s}$, distribution transforming encoders (DTEs) for all efficiently samplable distributions are infeasible. Even the computational relaxation of DTEs is infeasible assuming extractable one-way functions. Since all currently known constructions of honey encryption rely on DTEs, we conditionally refute the existence of honey encryption based on the DTE-then-encrypt framework from [JR14]. On the positive side, due to our feasibility result of $\mathsf{acPREH}^{\mathsf{rand}}_{\approx_c}$, computational honey encryption is feasible in the CRS model.

THEOREM (informal). *Assuming* $\mathsf{acPREH}^{\mathsf{rand}}_{\approx_c}$ *and a suitable symmetric-key encryption scheme (modeled as a random cipher), computational honey encryption for all efficiently samplable distributions exists in the CRS model.*

14.3 NEGATIVE RESULTS FOR STRONGER NOTIONS

Below we describe how we gradually relax perfect compression via different notions of pseudorandom encodings and derive infeasibility results for all variants of pseudorandom encodings which restrict the encoding algorithm to be deterministic or require an information-theoretic pseudorandomness guarantee. This leaves computational randomized pseudorandom encodings as the best possible achievable notion.

14.3.1 Deterministic, Statistical Pseudorandom Encodings

The notion of pseudorandom encodings with a deterministic encoding algorithm and information-theoretic indistinguishability is perhaps the simplest notion one can consider. As we will prove in this paper, this notion recovers the notion of perfect compression: since the encoding algorithm for some source X is deterministic, it can be seen with an entropy argument that the output size of E_X must be at most $\mathrm{H}_\infty(X)$, the min-entropy of X; otherwise, the distribution $\{\mathsf{E}_X(X)\}$ can necessarily be distinguished from random with some statistically non-negligible advantage. Therefore, E_X is a perfect and efficient compression algorithm for X, with decompression algorithm D_X; this is true even for the relaxation in the CRS model. The existence of efficient compression algorithms for various categories of samplers was thoroughly studied [TVZ05]. In particular, the existence of compression algorithms for all efficiently samplable sources implies the inexistence of one-way functions (this is an observation attributed to Levin in [GS85]) since compressing the output of a pseudorandom generator to its entropy would distinguish it from a random string, and the existence of one-way functions implies the existence of pseudorandom generators [HILL99].

THEOREM (informal). *Assuming the existence of one-way functions, then neither* $\mathsf{PREH}^{\mathsf{det}}_{\equiv_s}$ *nor* $\mathsf{cPREH}^{\mathsf{det}}_{\equiv_s}$ *hold.*

This is a strong impossibility result, as one-way functions dwell among the weakest assumptions in cryptography, [Imp95]. One can circumvent this impossibility by studying whether compression can be achieved for more restricted classes of distributions, as was done for instance in [TVZ05].

Our work can be seen as pursuing an orthogonal direction. We seek to determine whether a relaxed notion of compression can be achieved for *all* efficiently samplable distributions. The relaxations we consider comprise the possibility to use randomness in the encoding algorithm, and relaxing the requirement on the encoded distribution to be only computationally indistinguishable from random. Clearly, these relaxations remove one of the most important features of compression algorithms, which is that their outputs are smaller than their inputs (i. e., they compress). Nevertheless, the indistinguishability of the encoded distribution from the uniform distribution is another crucial feature of perfect compression algorithms, which has independent applications.

14.3.2 Deterministic, Computational Pseudorandom Encodings

We now turn towards a relaxation where the encoded distribution is only required to be computationally indistinguishable from random, but the encoding algorithm is still required to be deterministic. This flavor is strongly connected to an important problem in cryptography: the problem of separating HILL entropy [HILL99] from Yao entropy [Yao82]. HILL and Yao entropy are different approaches of formalizing computational entropy, i. e., the amount of entropy a distribution appears to have from the viewpoint of a computationally bounded entity. Informally, a distribution has high HILL entropy if it is computationally close to a distribution with high min-entropy; a distribution has high Yao entropy if it cannot be compressed efficiently. Finding a distribution which, under standard cryptographic assumptions, has high Yao entropy, but low HILL entropy constitutes a long standing open problem in cryptography. Currently, only an oracle separation [Wee04] and a separation for conditional distributions [HLR07] are known. To establish the connection between $\mathsf{PREH}^{\mathsf{det}}_{\approx_c}$ and this problem, we proceed as follows: informally, a deterministic pseudorandom encoding must necessarily *compress its input to the HILL entropy of the distribution*. That is, the output size of the encoding cannot be much larger than the HILL entropy of the distribution. This, in turn, implies that a distribution which admits such a pseudorandom encoding cannot have high Yao entropy.

In this work, we formalize the above argument, and show that the *conditional* separation of HILL and Yao entropy from [HLR07] suffices to refute $\mathsf{PREH}^{\mathsf{det}}_{\approx_c}$. This separation holds under the assumption that non-interactive zero-knowledge proofs with some appropriate structural properties exist (which in turn can be based on standard assumptions such as the quadratic residuosity assumption). Thus, we obtain the following infeasibility result:

THEOREM (informal). *If the quadratic residuosity assumption (QRA) holds, then* $\mathsf{PREH}^{\mathsf{det}}_{\approx_c}$ *does not hold.*

Hence, we may conclude that towards a feasible variant of pseudorandom encodings for all efficiently samplable distributions, requiring the encoding algorithm to be deterministic poses a strong restriction.

14.3.3 Randomized, Statistical Pseudorandom Encodings

We now consider the relaxation of perfect compression by allowing the encoding algorithm to be randomized while still requiring information-

theoretic indistinguishability from randomness. This flavor of pseudorandom encoding was used in the context of *honey encryption* [JR14]. Honey encryption is a cryptographic primitive which has been introduced to mitigate attacks on encryption schemes resulting from the use of low-entropy passwords. Honey encryption has the property that decrypting a ciphertext with an incorrect key always yields a valid-looking plaintext which seems to come from the expected distribution, thereby mitigating brute-force attacks. This is the same property that was useful in the previous PAKE example.

The study of honey encryption was initiated in [JR14], where it was shown that honey encryption can naturally be constructed by composing a block cipher (modeled as a random cipher) with a *distribution transforming encoder* (DTE), a notion which is equivalent to our notion of pseudorandom encoding with randomized encoding and statistical pseudorandomness. The focus of [JR14] was on obtaining such DTEs for simple and useful distributions. In contrast, we seek to understand the feasibility of this notion for *arbitrary* distributions. Intuitively, it is not straightforward to encode any efficient distribution into the uniform distribution; consider for example the distribution over RSA moduli, i. e., products of two random n-bit primes. Since no efficient algorithm is known to test membership in the support of this distribution, natural approaches seem to break down. In fact, we show in this work that this difficulty is inherent: building on techniques from [BCPR16; IKOS10], we demonstrate the impossibility of (randomized, statistical) pseudorandom encodings for all efficiently samplable distributions, under a relatively standard cryptographic assumption.

THEOREM (informal). *Assuming the subexponential hardness of the learning with errors (LWE) problem,* $\mathsf{PREH}^{\mathsf{rand}}_{\equiv_s}$ *does not hold.*

This result directly implies that under the same assumption, there exist efficiently samplable distributions (with input) for which no distribution transforming encoder exists. We view it as an interesting open problem whether this result can be extended to rule out the existence of honey encryption for arbitrary distributions under the same assumption.

14.4 OPEN QUESTIONS AND SUBSEQUENT WORK

The most intriguing question left open by our work is whether the weakest variant of the pseudorandom encoding hypothesis $\mathsf{cPREH}^{\mathsf{rand}}_{\approx_c}$, which is implied by IO, also *implies* IO. Very recently, this question was settled in the affirmative by Wee and Wichs [WW20] under the LWE assumption. More concretely, by modifying a heuristic IO construction of Brakerski et al. [BDGM20a], they show that IO is implied by LWE if one is additionally given an *oblivious LWE-sampler* in the CRS model. Such a sampler, given a matrix $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$, generates outputs that are indistinguishable from LWE samples $\mathbf{A} \cdot \mathbf{s} + \boldsymbol{e}$ without knowing the secrets $\mathbf{s}$ or the noise $\boldsymbol{e}$. The existence of an oblivious LWE sampler is nontrivial even under the LWE assumption, because $\mathbf{A}$ can be such that $\mathbf{A} \cdot \mathbf{s} + \boldsymbol{e}$ is not pseudorandom; however, such a sampler still follows from the invertible sampling hypothesis [IKOS10], which we show to be equivalent to the pseudorandom encoding hypothesis. By proposing an explicit heuristic construction of (a relaxed flavor of) an

oblivious LWE sampler, the end result of [WW20] is a construction of IO from a new "falsifiable" assumption.

Whether $\mathsf{cPREH}^{\mathsf{rand}}_{\approx_c}$ implies IO under weaker or different assumptions than LWE remains open. A potentially easier goal is using $\mathsf{cPREH}^{\mathsf{rand}}_{\approx_c}$ to construct public-key encryption from one-way functions. This is related to the possibility of constructing oblivious transfer from any public-key encryption in which public keys and ciphertexts are obliviously samplable [EGL85; GKM+00], which is implied by public-key encryption and $\mathsf{cPREH}^{\mathsf{rand}}_{\approx_c}$. Here $\mathsf{cPREH}^{\mathsf{rand}}_{\approx_c}$ is used to bypass the black-box separation between public-key encryption and oblivious transfer [GKM+00].

Finally, there is a lot of room for relaxing the intractability assumptions we use to rule out the statistical ($\mathsf{cPREH}^{\mathsf{rand}}_{\equiv_s}$) and deterministic ($\mathsf{cPREH}^{\mathsf{det}}_{\approx_c}$) flavors of pseudorandom encodings.

14.5 OVERVIEW OF TECHNIQUES

In this section, we elaborate on some of our technical results in more detail. In the following, we identify a PPT sampler S with the distribution (family) ensemble it induces.

THE RELATION TO INVERTIBLE SAMPLING. In this paragraph, we elaborate on the equivalence between pseudorandom encodings and invertible sampling. Recall that a PPT sampler S is *inverse samplable* [CFGN96; DN00; GKM+00], if there exists an alternative sampler $\overline{S}$ inducing a distribution which is computationally indistinguishable to the distribution induced by S such that the computations of $\overline{S}$ can be efficiently inverted. Efficiently inverting the computation of $\overline{S}$ means that there exists an efficient inverse sampler $\overline{S}^{-1}$ which, given an output of $\overline{S}$, recovers a well-distributed random tape for $\overline{S}$ to compute the given output in the following sense. The inverse sampled random tape is required to be computationally indistinguishable from the actually used random tape. More formally, a PPT sampler S is inverse samplable if there exists an efficient alternative sampler $\overline{S}$ and an efficient inverse sampler $\overline{S}^{-1}$ such that

$$\{y \leftarrow S(1^\lambda) \colon y\} \approx_c \{y \leftarrow \overline{S}(1^\lambda) \colon y\}, \tag{14.3}$$

$$\{y \leftarrow \overline{S}(1^\lambda; r) \colon (r, y)\} \approx_c \{y \leftarrow \overline{S}(1^\lambda) \colon (\overline{S}^{-1}(1^\lambda, y), y)\}. \tag{14.4}$$

We refer to Equation (14.3) as *closeness* and to Equation (14.4) as *invertibility*. If the sampler S admits an input m, the above is required to hold for all inputs m in the input space L, where $\overline{S}$ and $\overline{S}^{-1}$ both additionally receive m as input. In accordance with [IKOS10], we refer to the hypothesis that all PPT algorithms with input are inverse samplable as the *invertible sampling hypothesis*. Restricting the invertible sampling hypothesis to algorithms which do not admit inputs is denoted the *weak* invertible sampling hypothesis.

The concepts of inverse samplability and pseudorandom encodings are tightly connected. Suppose a PPT algorithm S is inverse samplable. Then, there exists an alternative and an inverse sampler $(\overline{S}, \overline{S}^{-1})$ satisfying closeness and invertibility. Invertibility guarantees that the inverse sampler $\overline{S}^{-1}$ on input of a sample y from $\overline{S}(1^\lambda)$, outputs a computationally well-distributed

random tape r. Hence, with overwhelming probability over the choice of $y \leftarrow \overline{S}(1^\lambda)$ and $r \leftarrow \overline{S}^{-1}(y)$, the alternative sampler on input of r, recovers y. In other words, the inverse sampler $\overline{S}^{-1}$ can be seen as encoding a given sample y, whereas the *de-randomized* alternative sampler $\overline{S}$ given this encoding *as random tape*, is able to recover y. Looking through the lens of pseudorandom encoding, this almost proves correctness except that y is sampled according to $\overline{S}(1^\lambda)$ instead of $S(1^\lambda)$. This difference can be bridged due to closeness. We now turn towards showing pseudorandomness of the encoded distribution. Due to closeness, the distributions $\{y \leftarrow \overline{S}(1^\lambda) \colon (\overline{S}^{-1}(1^\lambda, y), y)\}$ and $\{y \leftarrow S(1^\lambda) \colon (\overline{S}^{-1}(1^\lambda, y), y)\}$ are computationally indistinguishable. Invertibility guarantees that, given a sample y from $\overline{S}(1^\lambda)$, an encoding of y is indistinguishable to uniformly chosen randomness conditioned on the fact that decoding yields y. Removing y from this distribution almost corresponds to pseudorandomness, except that y is sampled according to $\overline{S}(1^\lambda)$ instead of $S(1^\lambda)$. Again, we are able to bridge this gap due to closeness. Note that we crucially use the fact that the initial randomness used by $\overline{S}$ resides outside of the view of an adversary. Summing up, if a PPT sampler S is inverse samplable, then it can be pseudorandomly encoded.

Interestingly, this connection turns out to be bidirectional. Suppose a PPT algorithm S can be pseudorandomly encoded. Then, there exists an efficient encoding algorithm E_S and an efficient deterministic decoding algorithm D_S satisfying correctness and pseudorandomness. Looking through the lens of invertible sampling, we identify the decoding algorithm to correspond to the alternative sampler (viewing the random tape of the alternative sampler as explicit input to D_S) and the encoding algorithm to correspond to the inverse sampler. Pseudorandomness guarantees that $\mathsf{E}_S(S(1^\lambda))$ is indistinguishable from uniform randomness. Hence, applying the decode algorithm D_S on uniform randomness is indistinguishable from applying D_S to outputs of $\mathsf{E}_S(S(1^\lambda))$. Correctness guarantees that $\mathsf{D}_S(\mathsf{E}_S(y))$ for y sampled according to $S(1^\lambda)$ recovers y with overwhelming probability. Thus, the distribution induced by applying D_S on uniform randomness is computationally close to the distribution induced by $S(1^\lambda)$. This shows closeness. For the purpose of arguing about invertibility, consider the distribution $A := \{y \leftarrow \mathsf{D}_S(r) \colon (r, y)\}$. Due to pseudorandomness r can be considered an encoded sample from $S(1^\lambda)$. Hence, A is indistinguishable to the distribution, where r is produced by $\mathsf{E}_S(y')$ for some *independent* $y' \leftarrow S(1^\lambda)$, i. e.,

$$\{y \leftarrow \mathsf{D}_S(r) \colon (r, y)\} \approx_c \{y' \leftarrow S(1^\lambda), r \leftarrow \mathsf{E}_S(y'), y \leftarrow \mathsf{D}_S(r) \colon (r, y)\}.$$

Note that by correctness, y and y' are identical with overwhelming probability. Therefore, A is indistinguishable to $\{y' \leftarrow S(1^\lambda), r \leftarrow \mathsf{E}_S(y') \colon (r, y')\}$. Since sampling y' via D_S applied on uniform randomness is computationally close to the above distribution due to closeness, invertibility follows. Summing up, a sampler S can be pseudorandomly encoded *if and only if* it is inverse samplable.

Likewise to the variations and relaxations described for pseudorandom encodings, we vary and relax the notion of invertible sampling. The inverse sampler can be required to be deterministic or allowed to be randomized. Further, closeness and invertibility can be required to hold information-theoretically or computationally. We denote these variants as $\mathsf{ISH}^{\mathsf{rand}}_{\approx_c}$, $\mathsf{ISH}^{\mathsf{rand}}_{\equiv_s}$, $\mathsf{ISH}^{\mathsf{det}}_{\approx_c}$ and $\mathsf{ISH}^{\mathsf{det}}_{\equiv_s}$. To circumvent impossibilities in the plain model, we also define

the relaxations in the common reference string model in static and adaptive flavors, denoted by the prefix "c" and "ac", respectively. The above equivalence extends to all introduced variations of the pseudorandom encoding and invertible sampling hypotheses.

THE STATIC-TO-ADAPTIVE TRANSFORMATION. The static variant of pseudorandom encodings in the CRS model only guarantees correctness and pseudorandomness as long as the input m for the sampler S is chosen independently of the CRS. The adaptive variant, on the other hand, provides correctness and pseudorandomness even for adaptive choices of inputs. Adaptive notions always imply their static analogues. Interestingly, for pseudorandom encodings, the opposite direction is true as well. The core idea is to use an *indirection* argument (similar to [HJK+16; CPR17; CPV17]) to delay CRS generation until during the actual encoding process. Thus, the advantage stemming from adaptively choosing the input is eliminated.

Suppose that the static variant of the pseudorandom encoding hypothesis in the CRS model is true and let S be some PPT sampler. Since S can be pseudorandomly encoded in the CRS model with static choice of inputs, there exist algorithms $(\mathsf{Setup}', \mathsf{E}', \mathsf{D}')$ such that static correctness and pseudorandomness hold. Further, the algorithm Setup' can also be pseudorandomly encoded as above. Let $(\mathsf{Setup}'', \mathsf{E}'', \mathsf{D}'')$ be the corresponding algorithms such that static correctness and pseudorandomness hold. Note that since the sampler Setup' does not expect an input, static and adaptive guarantees are equivalent.

Then, the sampler S can be pseudorandomly encoded in the CRS model with *adaptive* choice of inputs as follows. Initially, we sample a common reference string crs'' via $\mathsf{Setup}''(1^\lambda)$ and make it available to the parties. Given crs'' and a sample y from $S(m)$, adaptive encoding works in two phases. First, a fresh CRS crs' is sampled via $\mathsf{Setup}'(1^\lambda)$ and pseudorandomly encoded via $r_1 \leftarrow \mathsf{E}''(crs'', crs')$. Second, the given sample y is pseudorandomly encoded via $r_2 \leftarrow \mathsf{E}'(crs', m, y)$. The encoding of y then consists of (r_1, r_2). To decode, the CRS crs' is restored via $\mathsf{D}''(crs'', r_1)$. Then, using crs', the original sample y is recovered via $\mathsf{D}'(crs', m, r_2)$.

Since crs' is chosen freshly during the encoding process, the input m which may depend on crs'', cannot depend on crs'. Further, the distribution Setup'' does not expect an input. Hence, static guarantees suffice.

To realize that adaptive pseudorandomness holds, consider the encoding of $S(m)$ for some adaptively chosen message m. Since the view of $\mathcal{A}$ when choosing the message m is independent of crs', static pseudorandomness can be applied to replace the distribution $\mathsf{E}'(crs', m, S(m))$ with uniform randomness. Further, since the sampler Setup' does not expect any input, static pseudorandomness suffices to replace the distribution $\mathsf{E}''(crs'', \mathsf{Setup}'(1^\lambda))$ with uniform randomness. This proves adaptive pseudorandomness.

The adaptive variant of correctness follows similarly from the static variant of correctness. Consider the distribution of decoding an encoded sample of $S(m)$, where m is adaptively chosen. Since the sampler Setup' does not expect an input, static correctness can be applied to replace decoding $\mathsf{D}''(crs'', r_1)$ with the crs' sampled during encoding. Again, since crs' does not lie in the view of the adversary when choosing the message m, static correctness guarantees that decoding succeeds with overwhelming probability. This proves adaptive correctness.

DETERMINISTIC PSEUDORANDOM ENCODING AND COMPRESSION. The notion of pseudorandom encoding is inspired by the notion of compression. A tuple of deterministic functions $(\mathsf{E}_X, \mathsf{D}_X)$ is said to compress a source X_λ to length $m(\lambda)$ with decoding error $\epsilon(\lambda)$, if (i) $\Pr[\mathsf{D}_X(\mathsf{E}_X(X_\lambda)) \neq X_\lambda] \leqslant \epsilon(\lambda)$ and (ii) $\mathbb{E}[|\mathsf{E}_X(X_\lambda)|] \leqslant m(\lambda)$, see [Wee04; TVZ05]. Pseudorandom encoding partially recovers the notion of compression if we require the encoding algorithm to be deterministic. If a source X_λ can be pseudorandomly encoded with a deterministic encoding algorithm having output length $n(\lambda)$, then X_λ is compressible to length $n(\lambda)$. Note, however, that the converse direction is not true. Compression and decompression algorithms for a compressible source do not necessarily encode that source pseudorandomly. The output of a compression algorithm is not required to look pseudorandom and, in some cases, admits a specific structure which makes it easily distinguishable from uniform randomness, e. g., instances using Levin search, [TVZ05].

Clearly, the requirement for correctness poses a lower bound on the encoding length $n(\lambda)$ [Sha48]. Conversely, requiring the encoding algorithm E_X to be deterministic means that the only source of entropy in the distribution $\mathsf{E}_X(X_\lambda)$ originates from the source X_λ itself. Hence, for the distributions $\mathsf{E}_X(X_\lambda)$ and the uniform distribution over $\{0,1\}^{n(\lambda)}$ to be indistinguishable, the encoding length $n(\lambda)$ must be "sufficiently small". We observe that correctness together with the fact that E_X is deterministic implies that the event $\mathsf{E}_X(\mathsf{D}_X(\mathsf{E}_X(X_\lambda))) = \mathsf{E}_X(X_\lambda)$ occurs with overwhelming probability. Applying pseudorandomness yields that $\mathsf{E}_X(\mathsf{D}_X(U_{n(\lambda)})) = U_{n(\lambda)}$ holds with overwhelming probability, wherefore we can conclude that D_X operates almost injectively on the set $\{0,1\}^{n(\lambda)}$. Hence, the (smooth) min-entropy of $\mathsf{D}_X(U_{n(\lambda)})$ is at least $n(\lambda)$.

Considering information-theoretical pseudorandomness, the distributions $\mathsf{D}_X(U_{n(\lambda)})$ and X_λ are statistically close. Hence, by the reasoning above, the encoding length $n(\lambda)$ is upper bounded by the (smooth) min-entropy of the source X_λ. In conclusion, if a distribution can be pseudorandomly encoded such that the encoding algorithm is deterministic satisfying statistical pseudorandomness, then this distribution is compressible to its (smooth) min-entropy. Using a technical "Splitting Lemma", this extends to the relaxed variant of the pseudorandom encoding hypothesis in the CRS model.

Considering computational pseudorandomness, by a similar argument as above, we obtain that X_λ is computationally close to a distribution with min-entropy $n(\lambda)$. This does not yield a relation between the encoding length and the min-entropy of the source. However, we do obtain relations to computational analogues of entropy. Computational entropy is the amount of entropy a distribution appears to have from the perspective of a computationally bounded entity. The notion of HILL entropy [HILL99] is defined via the computational indistinguishability from a truly random distribution. More formally, a distribution X_λ has HILL entropy at least k, if there exists a distribution with min-entropy k which is computationally indistinguishable from X_λ. Hence, the encoding length $n(\lambda)$ is upper bounded by the HILL entropy of the source X_λ. Another important notion of computational entropy is the notion of Yao entropy [Yao82]. Yao entropy is defined via the incompressibility of a distribution. More precisely, a distribution X_λ has Yao entropy at least k if X_λ cannot be efficiently compressed to length less than k (and successfully decompressed). If a distribution can be pseudorandomly encoded with deterministic encoding, then it can be compressed to the en-

coding length $n(\lambda)$. This poses an upper bound on the Yao entropy of the source. In summary, this yields

$$n(\lambda) \leqslant H^{\mathsf{HILL}}(X_\lambda) \quad \text{and} \quad H^{\mathsf{Yao}}(X_\lambda) \leqslant n(\lambda). \tag{14.5}$$

However, due to [HLR07; LMS05], if the Quadratic Residuosity Assumption (QRA) is true, then there exist distributions which have low *conditional* HILL entropy while being *conditionally* incompressible, i. e., have high conditional Yao entropy.[59] The above observations, particularly Equation (14.5), can be extended to conditional HILL and conditional Yao entropy, by considering $\mathsf{PREH}^{\mathsf{det}}_{\approx_c}$ for PPT algorithms with input. Therefore, if the Quadratic Residuosity Assumption is true, $\mathsf{PREH}^{\mathsf{det}}_{\approx_c}$ cannot be true for those distributions.

Unfortunately, we do not know whether this extends to the relaxed variants of the pseudorandom encoding hypothesis admitting access to a CRS. On a high level, the problem is that the HILL entropy, in contrast to the min-entropy, does not remain untouched when additionally conditioning on some common reference string distribution, even though the initial distribution is independent of the CRS. Hence, the splitting technique cannot be applied here.

[59] *Let (X, Z) be a joint distribution. The conditional computational entropy is the entropy X appears to have to a bounded adversary when additionally given Z.*

15 THE PSEUDORANDOM ENCODING HYPOTHESIS

A source X is a probability distribution on strings. A family of sources is a probability ensemble $(X_\lambda)_{\lambda\in\mathbb{N}}$, where X_λ is distributed over $\{0,1\}^{k(\lambda)}$ for some polynomial p. A family of sources $(X_m)_{m\in L}$ can also be indexed by strings from some language $L \subseteq \{0,1\}^+$, where X_m is distributed over $\{0,1\}^{k(|m|)}$ for some polynomial p. We say that a source X_λ is efficiently samplable if there is a PPT algorithm S such that $S(1^\lambda)$ is distributed according to X_λ for all $\lambda \in \mathbb{N}$ (using random coins from $\{0,1\}^{p(\lambda)}$). We further say that a source X_m indexed by strings is efficiently samplable if there exists a PPT algorithm S such that $S(m)$ is distributed according to X_m for all $m \in L$ (using random coins from $\{0,1\}^{p(|m|)}$).

We initiate the study of the ability to encode efficiently samplable distributions into the uniform distribution. In the following, an efficiently samplable distribution will be defined by the corresponding sampler S with input space L. A distribution defined via S can be pseudorandomly encoded if there exists an efficient potentially randomized encoding algorithm E_S and an efficient deterministic decoding algorithm D_S such that for all $m \in L$, the probability for the event $D_S(E_S(S(m))) = S(m)$ is overwhelming and the distribution $E_S(S(m))$ is indistinguishable from the uniform distribution $U_{n(\lambda)}$. We work with the hypothesis that every efficiently samplable distribution can be pseudorandomly encoded. In this chapter, we formally define the pseudorandom encoding hypothesis and its variants.

DEFINITION 15.1 (Pseudorandom encoding hypothesis, $\mathsf{PREH}^{\mathsf{rand}}_{\approx_c}$). For every PPT algorithm S, there exist efficient algorithms E_S (the encoding algorithm) with output length $n(\lambda)$ and D_S (the decoding algorithm), where D_S is deterministic and E_S is randomized satisfying the following two properties.

CORRECTNESS. For all inputs $m \in L$,

$$\epsilon^{\mathsf{dec\text{-}error}}_{(E_S,D_S),m}(\lambda) \coloneqq \Pr\left[y \leftarrow S(m) \colon D_S(m, E_S(m,y)) \neq y\right]$$

is negligible.

PSEUDORANDOMNESS. For all PPT adversaries $\mathcal{A}$ and all inputs $m \in L$,

$$\mathrm{Adv}^{\mathsf{pre}}_{(E_S,D_S),\mathcal{A},m}(\lambda) \coloneqq \left| \Pr\left[\mathrm{Exp}^{(0)\text{-}\mathsf{pre}}_{(E_S,D_S),\mathcal{A},m}(\lambda) = 1\right] - \Pr\left[\mathrm{Exp}^{(1)\text{-}\mathsf{pre}}_{(E_S,D_S),\mathcal{A},m}(\lambda) = 1\right] \right|$$

is negligible, where $\mathrm{Exp}^{\mathsf{pre}}_{\mathcal{A},m,0}$ and $\mathrm{Exp}^{\mathsf{pre}}_{\mathcal{A},m,1}$ are defined in Figure 15.1.

REMARK 15.1. Definition 15.1 formulated for PPT algorithms S which do not admit an input m is called the *weak* $\mathsf{PREH}^{\mathsf{rand}}_{\approx_c}$.

DEFINITION 15.2 ($\mathsf{PREH}^{\mathsf{det}}_{\approx_c}$, $\mathsf{PREH}^{\mathsf{rand}}_{\equiv_s}$, $\mathsf{PREH}^{\mathsf{det}}_{\equiv_s}$). Definition 15.1 can be tuned in two dimensions: the encode algorithm can be required to be deterministic or allowed to be randomized, and the pseudorandomness property

$\mathrm{Exp}^{(0)\text{-pre}}_{(\mathsf{E}_S,\mathsf{D}_S),\mathcal{A},m}(\lambda)$

$r \leftarrow \{0,1\}^{p(\lambda)}$

$y \leftarrow S(m;r)$

return $\mathcal{A}(m, \mathsf{E}_S(m,y))$

(a) Pseudorandomness game using encoded samples.

$\mathrm{Exp}^{(1)\text{-pre}}_{(\mathsf{E}_S,\mathsf{D}_S),\mathcal{A},m}(\lambda)$

$u \leftarrow \{0,1\}^{n(\lambda)}$

return $\mathcal{A}(m,u)$

(b) Pseudorandomness game using true randomness.

Figure 15.1: The pseudorandomness games.

can be required to hold statistically or computationally. We denote these variants as $\mathsf{PREH}^{\beta}_{\alpha}$, where $\alpha \in \{\approx_c, \equiv_s\}$ and $\beta \in \{\mathsf{rand}, \mathsf{det}\}$.

REMARK 15.2. Definition 15.1 demands indistinguishability between encoded samples and the uniform distribution over all bitstrings of some length $n(\lambda)$. This requirement can be relaxed to indistinguishability from the uniform distribution over some efficiently samplable and efficiently recognizable set $\mathcal{R}$ of size N, where elements in $\mathcal{R}$ can be represented with $\mathcal{O}(\log N)$ bits.

DETERMINISTIC ENCODING AND COMPATIBLE SAMPLERS. Requiring the encoding algorithm E_S to be deterministic entails the existence of what we call "incompatible samplers" for which $\mathsf{PREH}^{\mathsf{det}}_{\equiv_s}$ and even $\mathsf{PREH}^{\mathsf{det}}_{\approx_c}$ are unconditionally false. For instance, consider the sampler S^* which on input of 1^λ uniformly chooses an element from the set $\{00, 01, 10\} \subset \{0,1\}^2$. Assume $\mathsf{PREH}^{\mathsf{det}}_{\approx_c}$ is true for S^*. Then, correctness requires that with overwhelming probability over the sampling process $y \leftarrow S^*(1^\lambda)$, $\mathsf{D}_{S^*}(\mathsf{E}_{S^*}(y)) = y$. Hence, E_{S^*} must map into the set $\{0,1\}^k$ for $k \geqslant 2$ (otherwise there is an correctness error of at least $\frac{1}{3}$). Pseudorandomness, on the other hand, requires that $\mathsf{E}_{S^*}(S^*(1^\lambda))$ is computationally indistinguishable from the uniform distribution over $\{0,1\}^k$. However, since E_{S^*} is a deterministic algorithm, $|\mathit{supp}(\mathsf{E}_{S^*}(S^*(1^\lambda)))| = 3$. Therefore, there exists at least one element in $\{0,1\}^k \setminus \mathit{supp}(\mathsf{E}_{S^*}(S^*(1^\lambda)))$. An adversary can easily determine this element by evaluating E_{S^*} on each element in the support of S^* (if the support of the sampler was super-polynomial, this would not be possible for a PPT adversary, but very well possible for an unbounded one).

Another example of such an incompatible sampler is a sampler with large support but low min-entropy (i. e., a sampler that has (at least one) very likely output and many much less likely outputs). For instance, consider the sampler S' with probability distribution

$$\Pr\left[S'(1^\lambda) = 0^\lambda\right] = \frac{1}{2},$$
$$\Pr\left[S'(1^\lambda) = 1 \,\|\, x\right] = \frac{1}{2^\lambda} \text{ for each } x \in \{0,1\}^{\lambda-1}.$$

Assume $\mathsf{PREH}^{\mathsf{det}}_{\approx_c}$ is true for S'. Then, correctness requires that an overwhelming fraction of the elements of the form $1 \,\|\, x$ for $x \in \{0,1\}^{\lambda-1}$ are correctly decodable. Furthermore, since the element 0^λ has a non-negligible probability to occur, it needs to be correctly decodable. Let the support of $\mathsf{E}_{S'}(S'(1^\lambda))$ be (a subset of) $\{0,1\}^{\lambda-1}$. Pseudorandomness requires that $\mathsf{E}_{S'}(S'(1^\lambda))$ is indistinguishable from the uniform distribution over $\{0,1\}^{\lambda-1}$. However, since

$\mathsf{E}_{S'}$ is deterministic, the value $\mathsf{E}_{S'}(0^\lambda)$ occurs with probability at least $\frac{1}{2}$ and, due to correctness, all other values $\mathsf{E}_{S'}(1 \| x)$ occur with far lower probability.

In order to obtain a meaningful definition of $\mathsf{PREH}^{\mathsf{det}}_{\approx_c}$ and $\mathsf{PREH}^{\mathsf{det}}_{\equiv_s}$, we restrict these hypotheses to only hold for specific classes which we refer to as "compatible samplers" $\mathcal{S}^{\approx_c}$ and $\mathcal{S}^{\equiv_s}$, respectively.

DEFINITION 15.3 (Compatibility with deterministic encodings). A sampler S is *statistically compatible with deterministic encodings* if there exists a set A whose cardinality is a power of 2, such that $\Pr[S(1^\lambda) \in A]$ is overwhelming, and S is ϵ-flat on A, i. e., for all $a \in A$ we have $|\Pr[S(1^\lambda) = a] - \frac{1}{|A|}| \leqslant \epsilon(\lambda)$, for some negligible function ϵ. The class $\mathcal{S}^{\equiv_s}$ contains all samplers which are statistically compatible with deterministic encodings.

A sampler S is *computationally compatible with deterministic encodings* if $S \in \mathcal{S}^{\equiv_s}$ *or* if $|supp(S(1^\lambda))|$ is super-polynomial and the min-entropy $H_\infty(S(1^\lambda)) \in \omega(\log(\lambda))$ (i. e., the most likely event occurs with negligible probability). The class $\mathcal{S}^{\approx_c}$ contains all samplers which are computationally compatible with deterministic encodings.

If S admits an input $z \in L$, S is statistically or computationally compatible with deterministic encodings if the corresponding criterion is met for all $z \in L$.

The above criterion for $\mathcal{S}^{\equiv_s}$ may seem unnatural. We note that by relaxing Definition 15.1 as noted in Remark 15.2, requiring high min-entropy suffices for a sampler to be statistically compatible with deterministic encodings.

Note that ϵ-flatness is a weaker criterion than statistical closeness to the uniform distribution over A. ϵ-flatness on A corresponds to closeness to the uniform distribution over A with respect to the infinity norm. Statistical closeness, however, is formalized with respect to the Manhattan norm. The deterministic pseudorandom encodings restricted to compatible samplers still has interesting connections.

Let iPRG be an injective PRG with stretch ℓ. Let S be the sampler which on input of 1^λ draws a uniform seed $s \in \{0,1\}^\lambda$ and outputs $\mathsf{iPRG}(s) \in \{0,1\}^{\ell(\lambda)}$. Clearly, S is statistically compatible with deterministic encodings.

LEMMA 15.1. *Let* PRG *be a PRG with stretch* ℓ *and let* S *be the sampler which on input of* 1^λ *produces the distribution* $\mathsf{PRG}(U_\lambda)$. *Then,* $S \in \mathcal{S}^{\equiv_s}$.

Proof. Let $A' := \mathsf{PRG}(\{0,1\}^\lambda)$ and let $A'' \subseteq \{0,1\}^{\ell(\lambda)} \setminus A'$ such that for $A := A' \cup A''$ we have $|A| = 2^\lambda$. We have $\Pr[S(1^\lambda) \in A] = 1$.

For flatness on A, we upper bound the probability of the most likely event in A. Due to pseudorandomness of PRG, all (non-uniform) PPT adversaries distinguishing the distributions $S(1^\lambda)$ and $U_{\ell(\lambda)}$ have a negligible advantage. Assume there exists an $a \in A'$, such that $\Pr[S(1^\lambda) = a] \geqslant \delta(\lambda)$, for a non-negligible function δ. Then, the most likely value in A' could be used as polynomial advice string allowing a non-uniform adversary to recognize the distribution $S(1^\lambda)$ with non-negligible probability $\delta(\lambda)$. (A uniform adversary can sample a random seed and evaluate the PRG. The thereby obtained output equals the most likely event a with probability $\delta(\lambda)$. Hence, a uniform adversary has advantage at least $\delta(\lambda)^2$ in distinguishing.) Therefore, for all $a \in A'$, $\Pr[S(1^\lambda) = a] \leqslant \epsilon(\lambda)$ for some negligible function ϵ. Hence, for all $a \in A$, we have $|\Pr[S(1^\lambda) = a] - \frac{1}{|A|}| \leqslant \epsilon(\lambda) + 2^{-\lambda}$ which is negligible. □

LEMMA 15.2. *Let* PRG *be a PRG with stretch* ℓ*. Let* S *be the sampler which on input of* $x \in L$ *produces the distribution* $\{y_1 \leftarrow \mathsf{PRG}(U_{|x|}), y_2 \leftarrow D_{x,y_1} : (y_1, y_2)\}$ *for any distribution* D*. Then,* $S \in \mathcal{S}^{\approx_c}$.

Proof. Let $\lambda := |x|$. Using the argument of Lemma 15.1, all bitstrings in $\mathsf{PRG}(\{0,1\}^\lambda)$ have only a negligible probability to occur. Therefore, for all $x \in L$ and all $(y_1, y_2) \in \mathit{supp}(S(x))$, we have $\Pr[S(1^\lambda, x) = (y_1, y_2)]$ is negligible. Therefore, $S \in \mathcal{S}^{\approx_c}$. □

IMPOSSIBILITY OF UNIVERSAL ENCODING. It is essential that the decoding algorithm D_S depends on the sampler S, since due to pseudorandomness, D_S on input of a random string needs to produce a sample that is in some sense close to the distribution produced by S. This argument does not hold necessarily for the encode algorithm. For instance, in the context of compression, the encoding algorithm does not necessarily depend on the distribution to be compressed. [TVZ05] study the notion of universal compression. This translates to the following definition of PREH with universal encoding.

DEFINITION 15.4 ($\mathsf{PREH}^\beta_\alpha$ with universal encoding). Let $\mathcal{S}$ be a class of sampling algorithms. We say $\mathsf{PREH}^\beta_\alpha$ *with universal sampling* is true for the class $\mathcal{S}$, if there exists a universal encoding algorithm E, such that for every PPT algorithm $S \in \mathcal{S}$, there exists an efficient deterministic decoding algorithm D_S, such that correctness and pseudorandomness are satisfied.

In contrast to universal compression, pseudorandom encoding with universal encoding is impossible.

LEMMA 15.3. *PREH with universal encoding is false.*

Proof. Consider the class of samplers $\mathcal{S} = \{S_1, S_2, S_3\}$ over $\{0,1\}^{k(\lambda)}$ with support $Y_{i,\lambda} := \mathit{supp}(S_i(1^\lambda))$. We require that for $i \in \{1,2\}$, $|Y_{i,\lambda}| = 2^{k+1}$, $|Y_{1,\lambda} \cap Y_{2,\lambda}| = 2^k$, $Y_{1,\lambda} \cap Y_{2,\lambda} = Y_{3,\lambda}$ for $k \in \mathcal{O}(\log\lambda)$, and that S_1, S_2 and S_3 produce uniform samples over their support (i. e., correspond to flat distributions). We note that $S_1, S_2, S_3 \in \mathcal{S}^{\equiv_s}$. For notational convenience we omit the dependency on λ in the following.

Let $\alpha \in \{\approx_c, \equiv_s\}$ and $\beta \in \{\mathsf{rand}, \mathsf{det}\}$. Assume toward a contradiction, that $\mathsf{PREH}^\beta_\alpha$ with universal encoding is true for the class $\mathcal{S}$ of sampling algorithms above. Let $\{0,1\}^{n(\lambda)}$ be the range of E. Due to correctness, for $i \in \{1,2\}$, there is a negligible function ϵ, such that

$$\begin{aligned}\Pr_{y \leftarrow Y_i}\left[D_{S_i}(E(y)) \neq y\right] &= \sum_{y' \in Y_i} \Pr_{y \leftarrow Y_i}\left[D_{S_i}(E(y)) \neq y \wedge y = y'\right] \\ &= \frac{1}{|Y_i|} \sum_{y' \in Y_i} \Pr_{y \leftarrow Y_i}\left[D_{S_i}(E(y)) \neq y \mid y = y'\right] \\ &\leqslant \epsilon(\lambda).\end{aligned}$$

Since $|Y_i|$ is polynomial, for each $y \in Y_i$, $\Pr[D_{S_i}(E(y)) = y] \geqslant 1 - \epsilon'(\lambda)$ for some negligible function ϵ', where the probability is solely over the randomness of E. Hence, for all $y \in Y_3 = Y_1 \cap Y_2$,

$$\Pr\left[u \leftarrow E(y) : y = D_{S_1}(u) = D_{S_2}(u) = D_{S_3}(u)\right] \geqslant 1 - \epsilon''(\lambda)$$

for some negligible function ϵ''.

Due to Theorem 16.1 (see Section 16.3), for $i \in \{1,2,3\}$, the distribution $\{u \leftarrow U_{n(\lambda)} : D_{S_i}(u)\}$ is computationally (statistically) close to the distribution produced by $S_i(1^\lambda)$. Hence, (up to some negligible fraction ϵ''') the algorithms D_{S_1} and D_{S_2} map at most half of the strings in $\{0,1\}^{n(\lambda)}$ into the same set $Y_1 \cap Y_2 = Y_3$.

We build an adversary $\mathcal{A}$ on pseudorandomness with respect to sampler S_3 as follows. On input of u, $\mathcal{A}$ outputs 1 if and only if $D_{S_1}(u) = D_{S_2}(u)$.

$$\Pr\left[\mathrm{Exp}^{(0)\text{-pre}}_{(E_S,D_S),\mathcal{A}}(\lambda) = 1\right] \geqslant 1 - \epsilon''(\lambda)$$
$$\Pr\left[\mathrm{Exp}^{(1)\text{-pre}}_{(E_S,D_S),\mathcal{A}}(\lambda) = 1\right] \leqslant \frac{1}{2} - \epsilon'''(\lambda)$$

Therefore, $\mathcal{A}$ has a non-negligible advantage $\mathrm{Adv}^{\mathsf{pre}}_{(E,D_{S_3}),\mathcal{A}}(\lambda)$. □

15.1 THE PSEUDORANDOM ENCODING HYPOTHESIS WITH SETUP

We obtain a natural relaxation of the pseudorandom encoding hypothesis by introducing public parameters. We refer to these public parameters as *global* or *non-programmable* common reference string. That is, a distribution defined via S can be pseudorandomly encoded in this relaxed sense, if there exists a probabilistic setup algorithm Setup_S and encode and decode algorithms as before such that for all $m \in L$, the event $D_S(crs, E_S(crs, S(m))) = S(m)$ is overwhelming, where the probability is also over the choice of crs, and the distribution $(\mathsf{Setup}_S(1^\lambda), E_S(\mathsf{Setup}_S(1^\lambda), S(m)))$ is indistinguishable from the distribution $(\mathsf{Setup}_S(1^\lambda), U_{n(\lambda)})$.

DEFINITION 15.5 ((Static) pseudorandom encoding hypothesis with setup, $\mathsf{cPREH}^{\mathsf{rand}}_{\approx_c}$). For every PPT algorithm S, there exist a PPT algorithm Setup_S and efficient algorithms (E_S, D_S), where D_S is deterministic and E_S is randomized (with output length $n(\lambda)$) satisfying the following two requirements.

CORRECTNESS. For all PPT adversaries $\mathcal{A}$,

$$\epsilon^{\mathsf{c\text{-}dec\text{-}error}}_{(\mathsf{Setup}_S,E_S,D_S),\mathcal{A}}(\lambda) \coloneqq \Pr\left[\begin{array}{l} m \leftarrow \mathcal{A}(1^\lambda) \\ crs \leftarrow \mathsf{Setup}_S(1^\lambda) : D_S(crs, m, E_S(crs, m, y)) \neq y \\ y \leftarrow S(m) \end{array}\right]$$

is negligible.

PSEUDORANDOMNESS. For all PPT adversaries $\mathcal{A}$,

$$\mathrm{Adv}^{\mathsf{crs\text{-}pre}}_{(\mathsf{Setup}_S,E_S,D_S),\mathcal{A}}(\lambda) \coloneqq \left|\Pr\left[\mathrm{Exp}^{(0)\text{-crs-pre}}_{(\mathsf{Setup}_S,E_S,D_S),\mathcal{A}}(\lambda) = 1\right] - \Pr\left[\mathrm{Exp}^{(0)\text{-crs-pre}}_{(\mathsf{Setup}_S,E_S,D_S),\mathcal{A}}(\lambda) = 1\right]\right|$$

is negligible, where $\mathrm{Exp}^{(0)\text{-crs-pre}}_{(\mathsf{Setup}_S,E_S,D_S),\mathcal{A}}(\lambda)$ and $\mathrm{Exp}^{(1)\text{-crs-pre}}_{(\mathsf{Setup}_S,E_S,D_S),\mathcal{A}}(\lambda)$ are defined in Figure 15.2.

$\mathrm{Exp}^{(0)\text{-crs-pre}}_{(\mathsf{Setup}_S,\mathsf{E}_S,\mathsf{D}_S),\mathcal{A}}(\lambda)$

$m \leftarrow \mathcal{A}(1^\lambda)$
$crs \leftarrow \mathsf{Setup}_S(1^\lambda)$
$r \leftarrow \{0,1\}^{p(\lambda)}$
$y \leftarrow S(m;r)$
return $\mathcal{A}(crs, m, \mathsf{E}_S(crs, m, y))$

(a) Pseudorandomness game using encoded samples.

$\mathrm{Exp}^{(1)\text{-crs-pre}}_{(\mathsf{Setup}_S,\mathsf{E}_S,\mathsf{D}_S),\mathcal{A}}(\lambda)$

$m \leftarrow \mathcal{A}(1^\lambda)$
$crs \leftarrow \mathsf{Setup}_S(1^\lambda)$
$u \leftarrow \{0,1\}^{n(\lambda)}$
return $\mathcal{A}(crs, m, u)$

(b) Pseudorandomness game using true randomness.

Figure 15.2: The static games experiments with setup.

We note that assuming non-uniform adversaries, correctness and pseudorandomness defined in Definition 15.5 can be equivalently defined by quantifying over all messages $m \in L$. Definition 15.5 is static in the sense that the inputs $m \in L$ are required to be chosen statically, i. e., independently of *crs*. In the following, we define the corresponding adaptive variant.

DEFINITION 15.6 (Adaptive pseudorandom encoding hypothesis with setup, $\mathsf{acPREH}^{\mathsf{rand}}_{\approx_c}$). For every PPT algorithm S, there exist a PPT algorithm Setup_S and efficient algorithms $(\mathsf{E}_S, \mathsf{D}_S)$, where D_S is deterministic and E_S is randomized (with output length $n(\lambda)$) satisfying the following two requirements.

CORRECTNESS. For all PPT adversaries $\mathcal{A}$,

$$\epsilon^{\mathsf{ac\text{-}dec\text{-}error}}_{(\mathsf{Setup}_S,\mathsf{E}_S,\mathsf{D}_S),\mathcal{A}}(\lambda) \coloneqq \Pr\left[\begin{array}{ll} crs \leftarrow \mathsf{Setup}_S(1^\lambda) & \\ m \leftarrow \mathcal{A}(crs) & : \mathsf{D}_S(crs, m, \mathsf{E}_S(crs, m, y)) \neq y \\ y \leftarrow S(m) & \end{array}\right]$$

is negligible.

PSEUDORANDOMNESS. For all PPT adversaries $\mathcal{A}$,

$$\mathrm{Adv}^{\mathsf{a\text{-}crs\text{-}pre}}_{(\mathsf{Setup}_S,\mathsf{E}_S,\mathsf{D}_S),\mathcal{A}}(\lambda) \coloneqq \left| \Pr\left[\mathrm{Exp}^{(0)\text{-a-crs-pre}}_{(\mathsf{Setup}_S,\mathsf{E}_S,\mathsf{D}_S),\mathcal{A}}(\lambda) = 1\right] - \Pr\left[\mathrm{Exp}^{(1)\text{-a-crs-pre}}_{(\mathsf{Setup}_S,\mathsf{E}_S,\mathsf{D}_S),\mathcal{A}}(\lambda) = 1\right] \right|$$

is negligible, where $\mathrm{Exp}^{(0)\text{-a-crs-pre}}_{(\mathsf{Setup}_S,\mathsf{E}_S,\mathsf{D}_S),\mathcal{A}}(\lambda)$ and $\mathrm{Exp}^{(1)\text{-a-crs-pre}}_{(\mathsf{Setup}_S,\mathsf{E}_S,\mathsf{D}_S),\mathcal{A}}(\lambda)$ are defined in Figure 15.3.

DEFINITION 15.7 ($\mathsf{cPREH}^{\mathsf{det}}_{\approx_c}$, $\mathsf{cPREH}^{\mathsf{rand}}_{\equiv_s}$, $\mathsf{cPREH}^{\mathsf{det}}_{\equiv_s}$, $\mathsf{acPREH}^{\mathsf{det}}_{\approx_c}$, $\mathsf{acPREH}^{\mathsf{rand}}_{\equiv_s}$, $\mathsf{acPREH}^{\mathsf{det}}_{\equiv_s}$). Definitions 15.5 and 15.6 can be tuned in two dimensions: the encoding algorithm can be required to be deterministic or allowed to be randomized, and the correctness and pseudorandomness properties can be required to hold statistically or computationally. We denote these variants as $\mathsf{cPREH}^{\beta}_{\alpha}$ and $\mathsf{acPREH}^{\beta}_{\alpha}$, respectively, where $\alpha \in \{\approx_c, \equiv_s\}$ and $\beta \in \{\mathsf{rand}, \mathsf{det}\}$.

REMARK 15.3 (Universal setup). In Definitions 15.5 and 15.6, we allow the algorithm Setup_S to depend on the sampler S. A natural (but somewhat incomparable) variant of this definition is to require the existence of a (bounded-

$\mathrm{Exp}^{(0)\text{-a-crs-pre}}_{(\mathsf{Setup}_S,\mathsf{E}_S,\mathsf{D}_S),\mathcal{A}}(\lambda)$

$crs \leftarrow \mathsf{Setup}_S(1^\lambda)$

$m \leftarrow \mathcal{A}(crs)$

$r \leftarrow \{0,1\}^{p(\lambda)}$

$y \leftarrow S(m;r)$

return $\mathcal{A}(crs, m, \mathsf{E}_S(crs, m, y))$

(a) Pseudorandomness game using encoded samples.

$\mathrm{Exp}^{(1)\text{-a-crs-pre}}_{(\mathsf{Setup}_S,\mathsf{E}_S,\mathsf{D}_S),\mathcal{A}}(\lambda)$

$crs \leftarrow \mathsf{Setup}_S(1^\lambda)$

$m \leftarrow \mathcal{A}(crs)$

$u \leftarrow \{0,1\}^{n(\lambda)}$

return $\mathcal{A}(crs, m, u)$

(b) Pseudorandomness game using true randomness.

Figure 15.3: The adaptive pseudorandomness games with setup.

circuit-size) *universal setup algorithm* $\mathsf{Setup}(1^\lambda, B)$ for $B \in \mathbb{N}$ which provides the above guarantees for all samplers which can be represented with B bits. We refer to this variant as universal $\mathsf{cPREH}^\beta_\alpha$ and universal $\mathsf{acPREH}^\beta_\alpha$, respectively.

REMARK 15.4 (Common random string). We will denote the strengthening of the pseudorandom encoding hypothesis as in Definitions 15.5 and 15.6, where the setup algorithm Setup_S is required to sample uniform random strings, as the pseudorandom encoding hypothesis with a *common random string* or *uniformly random CRS* (URC).

Note that in Definitions 15.5 and 15.6, we implicitly only consider *legitimate* adversaries which guarantee that $m \in L$. For the sake of avoiding notational overhead, we do not make this explicit.

15.2 STATIC-TO-ADAPTIVE TRANSFORMATION

Clearly, Definition 15.6 implies Definition 15.5. Interestingly, the opposite direction is true as well by an "indirection" argument similar to the one from the work on universal samplers [HJK+16]. A similar technique was also used in the context of non-committing encryption [CPR17] and adaptively secure MPC [CPV17].

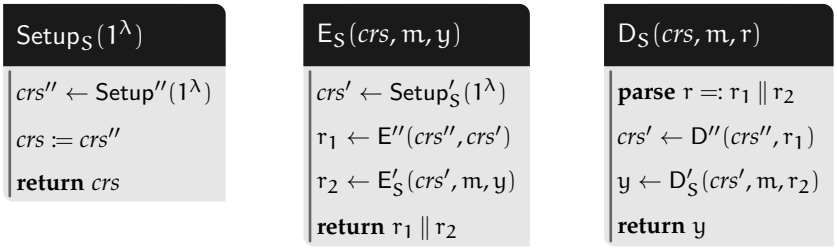
THEOREM 15.1. *Let* $\alpha \in \{\approx_c, \equiv_s\}$ *and* $\beta \in \{\mathsf{rand}, \mathsf{det}\}$. *If* $\mathsf{cPREH}^\beta_\alpha$ *is true, then* $\mathsf{acPREH}^\beta_\alpha$ *is true.*

Proof. We prove the statement for the computational randomized case. The proof directly extends to the remaining cases.

Let S be a PPT sampler with input space L. Since $\mathsf{cPREH}^{\mathsf{rand}}_{\approx_c}$ is true, for the PPT sampler S, there exist $(\mathsf{Setup}'_S, \mathsf{E}'_S, \mathsf{D}'_S)$ with output length $n'(\lambda)$ such that correctness and pseudorandomness hold (statically) as in Definition 15.5. Again, since $\mathsf{cPREH}^{\mathsf{rand}}_{\approx_c}$ is true, for the PPT sampler Setup'_S, there exist $(\mathsf{Setup}'', \mathsf{E}'', \mathsf{D}'')$ with output length $n''(\lambda)$ such that correctness and pseudorandomness hold (statically).[60] Note that Setup'_S does not expect an input.

In Figure 15.4, we define algorithms $(\mathsf{Setup}_S, \mathsf{E}_S, \mathsf{D}_S)$ satisfying *adaptive* correctness and pseudorandomness, as in Definition 15.6.

[60] *For notational convenience, we do not write the sampler* Setup'_S *as index.*



(a) Setup algorithm. **(b)** Encoding algorithm. **(c)** Decoding algorithm.

Figure 15.4: Adaptive pseudorandom encodings.

Intuitively, since crs' is chosen freshly and independently after the adversary fixes the message m, selective security suffices. Furthermore, since the distribution of crs' has no input, selective security is sufficient.

ADAPTIVE CORRECTNESS. We define a series of hybrid games to prove correctness, see Figure 15.5.

GAME $\mathbf{G}_0$. This game corresponds to encoding and decoding a sample from $S(m)$ for some adversarially and adaptively chosen message m. $\mathbf{G}_0$ outputs 1 if decoding succeeds.

GAME $\mathbf{G}_1$. $\mathbf{G}_1$ is identical to $\mathbf{G}_0$ except for some reordering. That is, the game hop from $\mathbf{G}_0$ to $\mathbf{G}_1$ is only conceptional and $\Pr[out_0 = 1] = \Pr[out_1 = 1]$.

GAME $\mathbf{G}_2$. $\mathbf{G}_2$ is identical to $\mathbf{G}_1$ except that for decoding y_D, $\mathbf{G}_2$ uses the originally sampled CRS crs' instead of the decoded CRS crs'_D.

CLAIM 15.1. For all PPT adversaries $\mathcal{A}$, there exists a PPT adversary $\mathcal{B}_2$, such that $|\Pr[out_2 = 1] - \Pr[out_1 = 1]| \leqslant \epsilon^{\text{c-dec-error}}_{(\mathsf{Setup}'', \mathsf{E}'', \mathsf{D}''), \mathcal{B}_2}(\lambda)$.

Proof. The games $\mathbf{G}_1$ and $\mathbf{G}_2$ proceed exactly identically if $crs'_D = crs'$. Let E be the event that $crs' \neq crs'_D$. We have that $out_1 = 1 \wedge \neg E \Leftrightarrow out_2 = 1 \wedge \neg E$. Due to correctness of $(\mathsf{Setup}'', \mathsf{E}'', \mathsf{D}'')$,

$$\Pr[E] = \Pr\left[\begin{array}{rcl} crs'' & \leftarrow & \mathsf{Setup}(1^\lambda) \\ crs' & \leftarrow & \mathsf{Setup}'_S(1^\lambda) \\ r_1 & \leftarrow & \mathsf{E}''(crs'', crs') \\ crs'_D & \leftarrow & \mathsf{D}''(crs'', r_1) \end{array} : crs'_D \neq crs'\right]$$

is negligible. Hence, the Difference Lemma (due to Shoup [Sho04]) upper bounds

$$\left|\Pr\left[out_2 = 1\right] - \Pr\left[out_1 = 1\right]\right| \leqslant \Pr[E].$$

□

GAME $\mathbf{G}_3$. $\mathbf{G}_3$ is identical to $\mathbf{G}_2$ except that r_1 and crs'_D are not produced and that crs' is sampled *after* the adversary chooses the message m (depending on crs''). Hence, the game hop from $\mathbf{G}_2$ to $\mathbf{G}_3$ is only conceptional and $\Pr[out_2 = 1] = \Pr[out_3 = 1]$.

CLAIM 15.2. For all PPT adversaries $\mathcal{A}$, there exists a PPT adversary $\mathcal{B}_3$, such that $\Pr[out_3 = 1] \geqslant 1 - \epsilon^{\text{c-dec-error}}_{(\mathsf{Setup}'_S, \mathsf{E}'_S, \mathsf{D}'_S), \mathcal{B}_3}(\lambda)$.

$\mathbf{G}_0$

$crs'' \leftarrow \mathsf{Setup}''(1^\lambda)$
$m \leftarrow \mathcal{A}(crs'')$
$y \leftarrow S(m)$
// encode
$crs' \leftarrow \mathsf{Setup}'_S(1^\lambda)$
$r_1 \leftarrow E''(crs'', crs')$
$r_2 \leftarrow E'_S(crs', m, y)$
// decode
$crs'_D \leftarrow D''(crs'', r_1)$
$y_D \leftarrow D'_S(crs'_D, m, r_2)$
return $y_D = y$

(a) Description of $\mathbf{G}_0$.

$\mathbf{G}_1$

$crs'' \leftarrow \mathsf{Setup}''(1^\lambda)$
$crs' \leftarrow \mathsf{Setup}'_S(1^\lambda)$
$r_1 \leftarrow E''(crs'', crs')$
$crs'_D \leftarrow D''(crs'', r_1)$
$m \leftarrow \mathcal{A}(crs'')$
$y \leftarrow S(m)$
$r_2 \leftarrow E'_S(crs', m, y)$
$y_D \leftarrow D'_S(crs'_D, m, r_2)$
return $y_D = y$

(b) Description of $\mathbf{G}_1$.

$\mathbf{G}_2$

$crs'' \leftarrow \mathsf{Setup}''(1^\lambda)$
$crs' \leftarrow \mathsf{Setup}'_S(1^\lambda)$
$r_1 \leftarrow E''(crs'', crs')$
$crs'_D \leftarrow D''(crs'', r_1)$
$m \leftarrow \mathcal{A}(crs'')$
$y \leftarrow S(m)$
$r_2 \leftarrow E'_S(crs', m, y)$
$y_D \leftarrow D'_S(crs', m, r_2)$
return $y_D = y$

(c) Description of $\mathbf{G}_2$.

$\mathbf{G}_3$

$crs'' \leftarrow \mathsf{Setup}''(1^\lambda)$
$m \leftarrow \mathcal{A}(crs'')$
$crs' \leftarrow \mathsf{Setup}'_S(1^\lambda)$
$y \leftarrow S(m)$
$r_2 \leftarrow E'_S(crs', m, y)$
$y_D \leftarrow D'_S(crs', m, r_2)$
return $y_D = y$

(d) Description of $\mathbf{G}_3$.

Figure 15.5: Hybrid games for the proof of adaptive correctness.

Proof. Due to correctness of $(\mathsf{Setup}'_S, E'_S, D'_S)$, we have that for all PPT adversaries $\overline{\mathcal{B}}$,

$$\Pr\left[\begin{array}{lcll} m & \leftarrow & \overline{\mathcal{B}}(1^\lambda) & \\ crs' & \leftarrow & \mathsf{Setup}'_S(1^\lambda) & \\ y & \leftarrow & S(m) & : y = y_D \\ r & \leftarrow & E'_S(crs', m, y) & \\ y_D & \leftarrow & D'_S(crs', m, r) & \end{array}\right]$$

is overwhelming. Consider the PPT adversary $\mathcal{B}_3$ who proceeds as in Figure 15.6.

Therefore, for all PPT adversaries $\mathcal{A}$, $\Pr[out_3 = 1]$ is overwhelming. □

Thus, we have

$$\epsilon^{\text{ac-dec-error}}_{(\mathsf{Setup}_S, E_S, D_S), \mathcal{A}}(\lambda) \leqslant \epsilon^{\text{c-dec-error}}_{(\mathsf{Setup}'', E'', D''), \overline{\mathcal{B}}}(\lambda) + \epsilon^{\text{c-dec-error}}_{(\mathsf{Setup}'_S, E'_S, D'_S), \overline{\mathcal{B}}'}(\lambda)$$

$\mathcal{B}_3(1^\lambda)$

$crs'' \leftarrow \mathsf{Setup}''(1^\lambda)$
$m \leftarrow \mathcal{A}(crs'')$
return m

Figure 15.6: Adversary $\mathcal{B}_3$ used for the game hop from $\mathbf{G}_2$ to $\mathbf{G}_3$.

for some PPT adversaries $\overline{\mathcal{B}}$ and $\overline{\mathcal{B}}'$.

ADAPTIVE PSEUDORANDOMNESS. We define a series of hybrid games to prove pseudorandomness, see Figure 15.7.

$\mathbf{G}_0$

$crs'' \leftarrow \mathsf{Setup}''(1^\lambda)$
$m \leftarrow \mathcal{A}(crs'')$
$y \leftarrow S(m)$
$crs' \leftarrow \mathsf{Setup}'_S(1^\lambda)$
$r_1 \leftarrow \mathsf{E}''(crs'', crs')$
$r_2 \leftarrow \mathsf{E}'_S(crs', m, y)$
return $\mathcal{A}(crs'', m, r_1 \| r_2)$

(a) Description of $\mathbf{G}_0$.

$\mathbf{G}_1$

$crs'' \leftarrow \mathsf{Setup}''(1^\lambda)$
$m \leftarrow \mathcal{A}(crs'')$
$y \leftarrow S(m)$
$crs' \leftarrow \mathsf{Setup}'_S(1^\lambda)$
$r_1 \leftarrow \mathsf{E}''(crs'', crs')$
$r_2 \leftarrow \{0,1\}^{n'(\lambda)}$
return $\mathcal{A}(crs'', m, r_1 \| r_2)$

(b) Description of $\mathbf{G}_1$.

$\mathbf{G}_2$

$crs'' \leftarrow \mathsf{Setup}''(1^\lambda)$
$crs' \leftarrow \mathsf{Setup}'_S(1^\lambda)$
$r_1 \leftarrow \mathsf{E}''(crs'', crs')$
$m \leftarrow \mathcal{A}(crs'')$
$r_2 \leftarrow \{0,1\}^{n'(\lambda)}$
return $\mathcal{A}(crs'', m, r_1 \| r_2)$

(c) Description of $\mathbf{G}_2$.

$\mathbf{G}_3$

$crs'' \leftarrow \mathsf{Setup}''(1^\lambda)$
$r_1 \leftarrow \{0,1\}^{n''(\lambda)}$
$m \leftarrow \mathcal{A}(crs'')$
$r_2 \leftarrow \{0,1\}^{n'(\lambda)}$
return $\mathcal{A}(crs'', m, r_1 \| r_2)$

(d) Description of $\mathbf{G}_3$.

Figure 15.7: Hybrid games for the proof of adaptive pseudorandomness.

GAME $\mathbf{G}_0$. $\mathbf{G}_0$ is identical to $\mathrm{Exp}^{(0)\text{-a-crs-pre}}_{(\mathsf{Setup}_S, \mathsf{E}_S, \mathsf{D}_S), \mathcal{A}}(\lambda)$.

GAME $\mathbf{G}_1$. $\mathbf{G}_1$ is identical to $\mathbf{G}_0$ except for the fact that in $\mathbf{G}_0$, r_2 is an encoding of y, whereas in $\mathbf{G}_1$, r_2 is a uniformly random bitstring.

CLAIM 15.3. For all PPT adversaries $\mathcal{A}$, there exists a PPT adversary $\mathcal{B}_1$, such that $|\Pr[out_1 = 1] - \Pr[out_0 = 1]| \leqslant \mathrm{Adv}^{\text{crs-pre}}_{(\mathsf{Setup}'_S, \mathsf{E}'_S, \mathsf{D}'_S), \mathcal{B}_1}(\lambda)$.

Proof. Construct an adversary $\mathcal{B}_1$ on *static* pseudorandomness relative to $(\mathsf{Setup}'_S, \mathsf{E}'_S, \mathsf{D}'_S)$ as follows. On input of 1^λ, $\mathcal{B}_1$ samples $crs'' \leftarrow \mathsf{Setup}''(1^\lambda)$, calls $\mathcal{A}$ on input of crs'', and outputs the message m produced by $\mathcal{A}$. In return, $\mathcal{B}_1$ receives $crs' \leftarrow \mathsf{Setup}'_S(1^\lambda)$ and either $u := \mathsf{E}'_S(crs', m, S(m))$ or

a uniform random string $u \leftarrow \{0,1\}^{n'(\lambda)}$ from $\mathrm{Exp}^{(b)\text{-crs-pre}}_{(\mathsf{Setup}'_S,\mathsf{E}'_S,\mathsf{D}'_S),\mathcal{A}}(\lambda)$. A_1 computes $r_1 \leftarrow \mathsf{E}''(crs'', crs')$, calls $\mathcal{A}$ on input of $(crs'', m, r_1 \| u)$ and returns $\mathcal{A}$'s output. See Figure 15.8 for a description of $\mathcal{B}_1$.

$\mathcal{B}_1(1^\lambda)$

$crs'' \leftarrow \mathsf{Setup}''(1^\lambda)$

$crs := crs''$

$m \leftarrow \mathcal{A}(crs'')$

return m

(a) First phase of $\mathcal{B}_1$.

$\mathcal{B}_1(crs', m, u)$

// $u \leftarrow \mathsf{E}'_S(crs', m, S(m))$ or $u \leftarrow \{0,1\}^{n'(\lambda)}$

$r_1 \leftarrow \mathsf{E}''(crs'', crs')$

return $\mathcal{A}(crs'', m, r_1 \| u)$

(b) Second phase of $\mathcal{B}_1$.

Figure 15.8: Adversary $\mathcal{B}_1$ used for the game hop from $\mathbf{G}_0$ to $\mathbf{G}_1$.

If $\mathcal{B}_1$ plays $\mathrm{Exp}^{(0)\text{-crs-pre}}_{(\mathsf{Setup}'_S,\mathsf{E}'_S,\mathsf{D}'_S),\mathcal{B}_1}(\lambda)$, then it perfectly simulates $\mathbf{G}_0$. On the other hand, if $\mathcal{B}_1$ plays $\mathrm{Exp}^{(1)\text{-crs-pre}}_{(\mathsf{Setup}'_S,\mathsf{E}'_S,\mathsf{D}'_S),\mathcal{B}_1}(\lambda)$, then it perfectly simulates $\mathbf{G}_1$. Hence, we have

$$\left|\Pr\left[out_1 = 1\right] - \Pr\left[out_0 = 1\right]\right| \leqslant \mathrm{Adv}^{\text{crs-pre}}_{(\mathsf{Setup}'_S,\mathsf{E}'_S,\mathsf{D}'_S),\mathcal{B}_1}(\lambda).$$

□

GAME $\mathbf{G}_2$. $\mathbf{G}_2$ is identical to $\mathbf{G}_1$ except for some reordering. More precisely, crs' and r_1 are sampled before $\mathcal{A}$ chooses the message m. The difference between $\mathbf{G}_1$ and $\mathbf{G}_2$ is only conceptional and $\Pr[out_2 = 1] = \Pr[out_1 = 1]$.

GAME $\mathbf{G}_3$. $\mathbf{G}_3$ is identical to $\mathbf{G}_2$ except that in $\mathbf{G}_2$, r_1 is produced as an encoding of crs' whereas in $\mathbf{G}_3$, r_1 is sampled uniformly at random. Hence, $\mathbf{G}_3$ is distributed identically to $\mathrm{Exp}^{(1)\text{-a-crs-pre}}_{(\mathsf{Setup}_S,\mathsf{E}_S,\mathsf{D}_S),\mathcal{A}}(\lambda)$.

CLAIM 15.4. For all PPT adversaries $\mathcal{A}$, there exists a PPT adversary $\mathcal{B}_3$, such that $|\Pr[out_3 = 1] - \Pr[out_2 = 1]| \leqslant \mathrm{Adv}^{\text{crs-pre}}_{(\mathsf{Setup}'',\mathsf{E}'',\mathsf{D}''),\mathcal{B}_3}(\lambda)$.

Proof. Construct an adversary $\mathcal{B}_3$ on *static* pseudorandomness relative to $(\mathsf{Setup}'', \mathsf{E}'', \mathsf{D}'')$ as follows. On input of 1^λ, $\mathcal{B}_3$ returns $\perp$ since the input space L of the sampler $\mathsf{Setup}'_S(1^\lambda)$ is empty. In return, $\mathcal{B}_3$ receives crs'' sampled via $\mathsf{Setup}''(1^\lambda)$ and u which is either produced via $\mathsf{E}''(crs'', \mathsf{Setup}'(1^\lambda))$ or via uniform sampling from $\{0,1\}^{n''(\lambda)}$. $\mathcal{B}_3$ calls $\mathcal{A}$ on input of crs'' and receives a message m from $\mathcal{A}$. Finally, $\mathcal{B}_3$ samples $r_2 \leftarrow \{0,1\}^{n'(\lambda)}$, calls $\mathcal{A}$ on input of $(crs'', m, u \| r_2)$ and outputs his output. See Figure 15.9 for a description of $\mathcal{B}_3$.

If $\mathcal{B}_3$ plays $\mathrm{Exp}^{(0)\text{-crs-pre}}_{(\mathsf{Setup}'',\mathsf{E}'',\mathsf{D}''),\mathcal{B}_3}(\lambda)$, then it perfectly simulates $\mathbf{G}_2$. On the other hand, if $\mathcal{B}_3$ plays $\mathrm{Exp}^{(1)\text{-crs-pre}}_{(\mathsf{Setup}'',\mathsf{E}'',\mathsf{D}''),\mathcal{B}_3}(\lambda)$, then it perfectly simulates $\mathbf{G}_3$. Hence, we have

$$\left|\Pr\left[out_3 = 1\right] - \Pr\left[out_2 = 1\right]\right| \leqslant \mathrm{Adv}^{\text{crs-pre}}_{(\mathsf{Setup}'',\mathsf{E}'',\mathsf{D}''),\mathcal{B}_3}(\lambda).$$

□

$\mathcal{B}_3(1^\lambda)$

return $\perp$

(a) First phase of $\mathcal{B}_3$.

$\mathcal{B}_3(crs'', \mathfrak{u})$

// $\mathfrak{u} \leftarrow \mathsf{E}''(crs'', \mathsf{Setup}'(1^\lambda))$ or $\mathfrak{u} \leftarrow \{0,1\}^{n''(\lambda)}$

$\mathfrak{m} \leftarrow \mathcal{A}(crs'')$

$r_2 \leftarrow \{0,1\}^{n''(\lambda)}$

return $\mathcal{A}(crs'', \mathfrak{m}, \mathfrak{u} \| r_2)$

(b) Second phase of $\mathcal{B}_3$.

Figure 15.9: Adversary $\mathcal{B}_3$ used for the game hop from $\mathbf{G}_2$ to $\mathbf{G}_3$.

Therefore, we have

$$\begin{aligned}\mathrm{Adv}^{\text{a-crs-pre}}_{(\mathsf{Setup}_S,\mathsf{E}_S,\mathsf{D}_S),\mathcal{A}}(\lambda) &= \left|\Pr\left[out_3 = 1\right] - \Pr\left[out_0 = 1\right]\right| \\ &\leqslant \mathrm{Adv}^{\text{crs-pre}}_{(\mathsf{Setup}'',\mathsf{E}'',\mathsf{D}''),\overline{\mathcal{B}}}(\lambda) \\ &\quad + \mathrm{Adv}^{\text{crs-pre}}_{(\mathsf{Setup}'_S,\mathsf{E}'_S,\mathsf{D}'_S),\overline{\mathcal{B}}'}(\lambda)\end{aligned}$$

for some PPT adversaries $\overline{\mathcal{B}}$ and $\overline{\mathcal{B}}'$. □

16 PSEUDORANDOM ENCODINGS AND INVERTIBLE SAMPLING

In this chapter, we explain the relation between pseudorandom encodings and invertible sampling, [DNo0]. In Sections 16.1 and 16.2, we restate the invertible sampling hypothesis of [IKOS10] and define several variants thereof. In Section 16.3, we prove that a distribution can be pseudorandomly encoded if and only if it is inverse samplable. This extends to all of the introduced variations of the pseudorandom encoding hypothesis and the invertible sampling hypothesis.

16.1 THE INVERTIBLE SAMPLING HYPOTHESIS

A PPT algorithm S is inverse samplable according to [DNo0; IKOS10] if there exists an alternative PPT algorithm $\overline{S}$ and a corresponding inverse sampler $\overline{S}^{-1}$ such that $\overline{S}$ (on every input) induces a distribution which is computationally indistinguishable from the distribution induced by S (on identical inputs) and $\overline{S}^{-1}$ inverses the computation of $\overline{S}$. That is, $\overline{S}^{-1}$ on input of an output produced by $\overline{S}$ produces computationally well-distributed random coins for $\overline{S}$ to produce the given output.

DEFINITION 16.1 (Invertible sampling hypothesis, $\mathsf{ISH}^{\mathsf{rand}}_{\approx_c}$, [IKOS10]). For every PPT algorithm S, there exists a PPT algorithm $\overline{S}$ (the alternate sampler) with randomness space $\{0,1\}^{n(\lambda)}$ and an efficient randomized algorithm $\overline{S}^{-1}$ (the inverse sampler), satisfying the following two properties.

CLOSENESS. For all PPT adversaries $\mathcal{A}$ and all inputs $m \in L$,

$$\mathrm{Adv}^{\mathsf{close}}_{(\overline{S},\overline{S}^{-1}),\mathcal{A},m}(\lambda) := \left| \Pr\left[\mathrm{Exp}^{(0)\text{-}\mathsf{close}}_{(\overline{S},\overline{S}^{-1}),\mathcal{A},m}(\lambda) = 1\right] - \Pr\left[\mathrm{Exp}^{(1)\text{-}\mathsf{close}}_{(\overline{S},\overline{S}^{-1}),\mathcal{A},m}(\lambda) = 1\right] \right|$$

is negligible, where $\mathrm{Exp}^{(0)\text{-}\mathsf{close}}_{(\overline{S},\overline{S}^{-1}),\mathcal{A},m}(\lambda)$ and $\mathrm{Exp}^{(1)\text{-}\mathsf{close}}_{(\overline{S},\overline{S}^{-1}),\mathcal{A},m}(\lambda)$ are defined in Figure 16.1.

INVERTIBILITY. For all PPT adversaries $\mathcal{A}$ and all inputs $m \in L$,

$$\mathrm{Adv}^{\mathsf{inv}}_{(\overline{S},\overline{S}^{-1}),\mathcal{A},m}(\lambda) := \left| \Pr\left[\mathrm{Exp}^{(0)\text{-}\mathsf{inv}}_{(\overline{S},\overline{S}^{-1}),\mathcal{A},m}(\lambda) = 1\right] - \Pr\left[\mathrm{Exp}^{(1)\text{-}\mathsf{inv}}_{(\overline{S},\overline{S}^{-1}),\mathcal{A},m}(\lambda) = 1\right] \right|$$

is negligible, where $\mathrm{Exp}^{(0)\text{-}\mathsf{inv}}_{(\overline{S},\overline{S}^{-1}),\mathcal{A},m}(\lambda)$ and $\mathrm{Exp}^{(1)\text{-}\mathsf{inv}}_{(\overline{S},\overline{S}^{-1}),\mathcal{A},m}(\lambda)$ are defined in Figure 16.1.

$\mathrm{Exp}^{(0)\text{-close}}_{(\overline{S},\overline{S}^{-1}),\mathcal{A},m}(\lambda)$

$r \leftarrow \{0,1\}^{p(\lambda)}$

$y \leftarrow S(m;r)$

return $\mathcal{A}(m,y)$

(a) Closeness game using the original sampler.

$\mathrm{Exp}^{(1)\text{-close}}_{(\overline{S},\overline{S}^{-1}),\mathcal{A},m}(\lambda)$

$r \leftarrow \{0,1\}^{n(\lambda)}$

$y \leftarrow \overline{S}(m;r)$

return $\mathcal{A}(m,y)$

(b) Closeness game using the alternative sampler.

$\mathrm{Exp}^{(0)\text{-inv}}_{(\overline{S},\overline{S}^{-1}),\mathcal{A},m}(\lambda)$

$r \leftarrow \{0,1\}^{n(\lambda)}$

$y \leftarrow \overline{S}(m;r)$

return $\mathcal{A}(m,r,y)$

(c) Invertibility game using the actually used randomness.

$\mathrm{Exp}^{(1)\text{-inv}}_{(\overline{S},\overline{S}^{-1}),\mathcal{A},m}(\lambda)$

$r \leftarrow \{0,1\}^{n(\lambda)}$

$y \leftarrow \overline{S}(m;r)$

$\overline{r} \leftarrow \overline{S}^{-1}(m,y)$

return $\mathcal{A}(m,\overline{r},y)$

(d) Invertibility game using the inverse sampled randomness.

Figure 16.1: The closeness and invertibility experiments.

DEFINITION 16.2 ($\mathsf{ISH}^{\mathsf{det}}_{\approx_c}$, $\mathsf{ISH}^{\mathsf{rand}}_{\equiv_s}$, $\mathsf{ISH}^{\mathsf{det}}_{\equiv_s}$). Definition 16.1 can be tuned in two dimensions: the inverse sampler can be required to be deterministic or allowed to be randomized, and the closeness and invertibility properties can be required to hold statistically or computationally. We denote these variants as $\mathsf{ISH}^{\beta}_{\alpha}$, where $\alpha \in \{\approx_c, \equiv_s\}$ and $\beta \in \{\mathsf{rand}, \mathsf{det}\}$.

16.2 THE INVERTIBLE SAMPLING HYPOTHESIS WITH SETUP

The invertible sampling hypothesis can be naturally relaxed by introducing public parameters (or global CRS), henceforth denoted *crs*. This allows the alternative sampler and the inverse sampler to use *crs*. Closeness and invertibility are defined against adversaries knowing the *crs* but choosing the inputs statically.

DEFINITION 16.3 (Invertible sampling hypothesis with setup, $\mathsf{cISH}^{\mathsf{rand}}_{\approx_c}$). For every PPT algorithm S there exists a PPT algorithm Setup_S, a PPT algorithm $\overline{S}$ (with randomness space $\{0,1\}^{n(\lambda)}$) and an efficient randomized $\overline{S}^{-1}$ satisfying the following two properties.

CLOSENESS. For all PPT adversaries $\mathcal{A}$,

$$\mathrm{Adv}^{\text{crs-close}}_{(\mathsf{Setup}_S,\overline{S},\overline{S}^{-1}),\mathcal{A}}(\lambda) \coloneqq \left| \Pr\left[\mathrm{Exp}^{(0)\text{-crs-close}}_{(\mathsf{Setup}_S,\overline{S},\overline{S}^{-1}),\mathcal{A}}(\lambda) = 1\right] - \Pr\left[\mathrm{Exp}^{(1)\text{-crs-close}}_{(\mathsf{Setup}_S,\overline{S},\overline{S}^{-1}),\mathcal{A}}(\lambda) = 1\right] \right|$$

is negligible, where $\mathrm{Exp}^{(0)\text{-crs-close}}_{(\mathsf{Setup}_S,\overline{S},\overline{S}^{-1}),\mathcal{A}}(\lambda)$ and $\mathrm{Exp}^{(1)\text{-crs-close}}_{(\mathsf{Setup}_S,\overline{S},\overline{S}^{-1}),\mathcal{A}}(\lambda)$ are defined in Figure 16.2.

INVERTIBILITY. For all PPT adversaries $\mathcal{A}$,

$$\mathsf{Adv}^{\text{crs-inv}}_{(\mathsf{Setup}_S,\overline{S},\overline{S}^{-1}),\mathcal{A}}(\lambda) := \left| \Pr\left[\mathsf{Exp}^{(0)\text{-crs-inv}}_{(\mathsf{Setup}_S,\overline{S},\overline{S}^{-1}),\mathcal{A}}(\lambda) = 1\right] - \Pr\left[\mathsf{Exp}^{(1)\text{-crs-inv}}_{(\mathsf{Setup}_S,\overline{S},\overline{S}^{-1}),\mathcal{A}}(\lambda) = 1\right] \right|$$

is negligible, where $\mathsf{Exp}^{(0)\text{-crs-inv}}_{(\mathsf{Setup}_S,\overline{S},\overline{S}^{-1}),\mathcal{A}}(\lambda)$ and $\mathsf{Exp}^{(1)\text{-crs-inv}}_{(\mathsf{Setup}_S,\overline{S},\overline{S}^{-1}),\mathcal{A}}(\lambda)$ are defined in Figure 16.2.

$\mathsf{Exp}^{(0)\text{-crs-close}}_{(\mathsf{Setup}_S,\overline{S},\overline{S}^{-1}),\mathcal{A}}(\lambda)$

$m \leftarrow \mathcal{A}(1^\lambda)$
$crs \leftarrow \mathsf{Setup}_S(1^\lambda)$
$r \leftarrow \{0,1\}^{p(\lambda)}$
$y \leftarrow S(m;r)$
return $\mathcal{A}(crs, y)$

(a) Closeness game using the original sampler.

$\mathsf{Exp}^{(1)\text{-crs-close}}_{(\mathsf{Setup}_S,\overline{S},\overline{S}^{-1}),\mathcal{A}}(\lambda)$

$m \leftarrow \mathcal{A}(1^\lambda)$
$crs \leftarrow \mathsf{Setup}_S(1^\lambda)$
$r \leftarrow \{0,1\}^{n(\lambda)}$
$y \leftarrow \overline{S}(crs, m;r)$
return $\mathcal{A}(crs, y)$

(b) Closeness game using the alternative sampler.

$\mathsf{Exp}^{(0)\text{-crs-inv}}_{(\mathsf{Setup}_S,\overline{S},\overline{S}^{-1}),\mathcal{A}}(\lambda)$

$m \leftarrow \mathcal{A}(1^\lambda)$
$crs \leftarrow \mathsf{Setup}_S(1^\lambda)$
$r \leftarrow \{0,1\}^{n(\lambda)}$
$y \leftarrow \overline{S}(crs, m;r)$
return $\mathcal{A}(crs, r, y)$

(c) Invertibility game using the actually used randomness.

$\mathsf{Exp}^{(1)\text{-crs-inv}}_{(\mathsf{Setup}_S,\overline{S},\overline{S}^{-1}),\mathcal{A}}(\lambda)$

$m \leftarrow \mathcal{A}(1^\lambda)$
$crs \leftarrow \mathsf{Setup}_S(1^\lambda)$
$r \leftarrow \{0,1\}^{n(\lambda)}$
$y \leftarrow \overline{S}(crs, m;r)$
$\overline{r} \leftarrow \overline{S}^{-1}(crs, m, y)$
return $\mathcal{A}(crs, \overline{r}, y)$

(d) Invertibility game using the inverse sampled randomness.

Figure 16.2: The static closeness and invertibility experiments with setup.

Definition 16.3 is static in the sense that closeness and invertibility adversaries are required to statically choose the challenge input $m \in L$. In the following, we define the corresponding adaptive version, where adversaries are allowed to choose the challenge input $m \in L$ depending on *crs*.

DEFINITION 16.4 (Adaptively secure invertible sampling hypothesis with setup, $\mathsf{aclSH}^{\mathsf{rand}}_{\approx_c}$). For every PPT algorithm S, there exists a PPT algorithm Setup_S, a PPT algorithm $\overline{S}$ (with randomness space $\{0,1\}^{n(\lambda)}$) and an efficient randomized $\overline{S}^{-1}$ satisfying the following two properties.

(ADAPTIVE) CLOSENESS. For all PPT adversaries $\mathcal{A}$,

$$\mathsf{Adv}^{\text{a-crs-close}}_{(\mathsf{Setup}_S,\overline{S},\overline{S}^{-1}),\mathcal{A}}(\lambda) := \left| \Pr\left[\mathsf{Exp}^{(0)\text{-a-crs-close}}_{(\mathsf{Setup}_S,\overline{S},\overline{S}^{-1}),\mathcal{A}}(\lambda) = 1\right] - \Pr\left[\mathsf{Exp}^{(1)\text{-a-crs-close}}_{(\mathsf{Setup}_S,\overline{S},\overline{S}^{-1}),\mathcal{A}}(\lambda) = 1\right] \right|$$

is negligible, where $\mathrm{Exp}^{(0)\text{-a-crs-close}}_{(\mathsf{Setup}_S,\overline{S},\overline{S}^{-1}),\mathcal{A}}(\lambda)$ and $\mathrm{Exp}^{(1)\text{-a-crs-close}}_{(\mathsf{Setup}_S,\overline{S},\overline{S}^{-1}),\mathcal{A}}(\lambda)$ are defined in Figure 16.3.

(ADAPTIVE) INVERTIBILITY. For all PPT adversaries $\mathcal{A}$,

$$\mathrm{Adv}^{\text{a-crs-inv}}_{(\mathsf{Setup}_S,\overline{S},\overline{S}^{-1}),\mathcal{A}}(\lambda) \coloneqq \left| \Pr\left[\mathrm{Exp}^{(0)\text{-a-crs-inv}}_{(\mathsf{Setup}_S,\overline{S},\overline{S}^{-1}),\mathcal{A}}(\lambda) = 1\right] - \Pr\left[\mathrm{Exp}^{(1)\text{-a-crs-inv}}_{(\mathsf{Setup}_S,\overline{S},\overline{S}^{-1}),\mathcal{A}}(\lambda) = 1\right] \right|$$

is negligible, where $\mathrm{Exp}^{(0)\text{-a-crs-inv}}_{(\mathsf{Setup}_S,\overline{S},\overline{S}^{-1}),\mathcal{A}}(\lambda)$ and $\mathrm{Exp}^{(1)\text{-a-crs-inv}}_{(\mathsf{Setup}_S,\overline{S},\overline{S}^{-1}),\mathcal{A}}(\lambda)$ are defined in Figure 16.3.

$\mathrm{Exp}^{(0)\text{-a-crs-close}}_{(\mathsf{Setup}_S,\overline{S},\overline{S}^{-1}),\mathcal{A}}(\lambda)$

$crs \leftarrow \mathsf{Setup}_S(1^\lambda)$
$m \leftarrow \mathcal{A}(crs)$
$r \leftarrow \{0,1\}^{p(\lambda)}$
$y \leftarrow S(m;r)$
return $\mathcal{A}(y)$

(a) Closeness game using the original sampler.

$\mathrm{Exp}^{(1)\text{-a-crs-close}}_{(\mathsf{Setup}_S,\overline{S},\overline{S}^{-1}),\mathcal{A}}(\lambda)$

$crs \leftarrow \mathsf{Setup}_S(1^\lambda)$
$m \leftarrow \mathcal{A}(crs)$
$r \leftarrow \{0,1\}^{n(\lambda)}$
$y \leftarrow \overline{S}(crs,m;r)$
return $\mathcal{A}(y)$

(b) Closeness game using the alternative sampler.

$\mathrm{Exp}^{(0)\text{-a-crs-inv}}_{(\mathsf{Setup}_S,\overline{S},\overline{S}^{-1}),\mathcal{A}}(\lambda)$

$crs \leftarrow \mathsf{Setup}_S(1^\lambda)$
$m \leftarrow \mathcal{A}(crs)$
$r \leftarrow \{0,1\}^{n(\lambda)}$
$y \leftarrow \overline{S}(crs,m;r)$
return $\mathcal{A}(r,y)$

(c) Invertibility game using the actually used randomness.

$\mathrm{Exp}^{(1)\text{-a-crs-inv}}_{(\mathsf{Setup}_S,\overline{S},\overline{S}^{-1}),\mathcal{A}}(\lambda)$

$crs \leftarrow \mathsf{Setup}_S(1^\lambda)$
$m \leftarrow \mathcal{A}(crs)$
$r \leftarrow \{0,1\}^{n(\lambda)}$
$y \leftarrow \overline{S}(crs,m;r)$
$\overline{r} \leftarrow \overline{S}^{-1}(crs,m,y)$
return $\mathcal{A}(\overline{r},y)$

(d) Invertibility game using the inverse sampled randomness.

Figure 16.3: The adaptive closeness and invertibility experiments with setup.

DEFINITION 16.5 ($\mathsf{cISH}^{\mathsf{det}}_{\approx_c}$, $\mathsf{cISH}^{\mathsf{rand}}_{\equiv_s}$, $\mathsf{cISH}^{\mathsf{det}}_{\equiv_s}$, $\mathsf{acISH}^{\mathsf{det}}_{\approx_c}$, $\mathsf{acISH}^{\mathsf{rand}}_{\equiv_s}$, $\mathsf{acISH}^{\mathsf{det}}_{\equiv_s}$). Definitions 16.3 and 16.4 can be tuned in two dimensions: the inverse sampler can be required to be deterministic or allowed to be randomized, and the closeness and invertibility properties can be required to hold statistically or computationally. We denote these variants as $\mathsf{cISH}^{\beta}_{\alpha}$ and $\mathsf{acISH}^{\beta}_{\alpha}$, respectively, where $\alpha \in \{\approx_c, \equiv_s\}$ and $\beta \in \{\mathsf{rand}, \mathsf{det}\}$.

REMARK 16.1 (Universal setup). In Definitions 16.3 and 16.4, we allow the algorithm Setup_S to depend on the sampler S. A natural (but somewhat incomparable) variant of this definition is to require the existence of a (bounded-circuit-size) *universal setup algorithm* $\mathsf{Setup}(1^\lambda, B)$ for $B \in \mathbb{N}$ which provides the above guarantees for all samplers which can be represented with B bits. We refer to this variant as universal $\mathsf{cISH}^{\beta}_{\alpha}$ and universal $\mathsf{acISH}^{\beta}_{\alpha}$, respectively.

REMARK 16.2 (Common random string). We will denote the strengthening of the invertible sampling hypothesis as in Definitions 16.3 and 16.4, where the setup algorithm Setup_S is required to sample uniform random strings, as the invertible sampling hypothesis with a *common random string* or *uniformly random CRS* (URC).

Again, in Definitions 16.3 and 16.4, we implicitly only consider *legitimate* adversaries which guarantee that $m \in L$.

16.3 EQUIVALENCE OF PSEUDORANDOM ENCODINGS AND INVERTIBLE SAMPLING

We prove the following theorem.

THEOREM 16.1. *Let* $\alpha \in \{\approx_c, \equiv_s\}$ *and* $\beta \in \{\mathsf{rand}, \mathsf{det}\}$. $\mathsf{PREH}^{\beta}_{\alpha}$ *is true if and only if* $\mathsf{ISH}^{\beta}_{\alpha}$ *is true.*

It is straight forward to extend Theorem 16.1 to the non-adaptive and adaptive variants with setup.

16.3.1 Every Inverse Samplable Distribution can be Pseudorandomly Encoded

LEMMA 16.1. *Let* $\alpha \in \{\approx_c, \equiv_s\}$ *and* $\beta \in \{\mathsf{rand}, \mathsf{det}\}$. *If* $\mathsf{ISH}^{\beta}_{\alpha}$ *holds, then* $\mathsf{PREH}^{\beta}_{\alpha}$ *holds.*

Proof. We prove this for the computational randomized case. The remaining cases are similar.

Assume $\mathsf{ISH}^{\mathsf{rand}}_{\approx_c}$ holds. Let S be a PPT algorithm. $\mathsf{ISH}^{\mathsf{rand}}_{\approx_c}$ implies that there exists an alternative sampler $\overline{S}$ (with randomness space $\{0,1\}^{n(\lambda)}$) and a corresponding inverse sampler $\overline{S}^{-1}$ satisfying closeness and invertibility.

For $m \in L$, $y \in \{0,1\}^*$, $r \in \{0,1\}^{n(\lambda)}$, we define the encode and decode algorithms as follows:

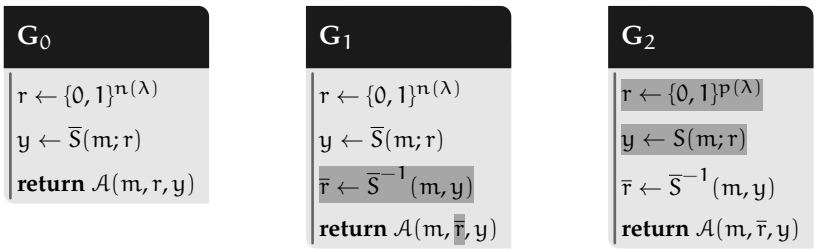
$$\mathsf{E}_S(m, y) \coloneqq \overline{S}^{-1}(m, y),$$
$$\mathsf{D}_S(m, r) \coloneqq \overline{S}(m; r).$$

CORRECTNESS. We consider a series of hybrids, see Figure 16.4. In the first game, $\mathsf{D}_S(m, r) = \overline{S}(m; r) = y$. Since all games are computationally close, this must hold for the last game as well.

GAME $\mathbf{G}_0$. $\mathbf{G}_0$ is identical to $\mathrm{Exp}^{(0)\text{-inv}}_{(\overline{S},\overline{S}^{-1}),\mathcal{A},m}(\lambda)$.

GAME $\mathbf{G}_1$. $\mathbf{G}_1$ is identical to $\mathrm{Exp}^{(1)\text{-inv}}_{(\overline{S},\overline{S}^{-1}),\mathcal{A},m}(\lambda)$. Hence, we have $|\Pr[out_1 = 1] - \Pr[out_0 = 1]| \leqslant \mathrm{Adv}^{\mathsf{inv}}_{(\overline{S},\overline{S}^{-1}),\mathcal{A},m}(\lambda)$.

GAME $\mathbf{G}_2$. $\mathbf{G}_2$ is identical to $\mathbf{G}_1$ except that in $\mathbf{G}_1$ y is sampled using the alternative sampler whereas in $\mathbf{G}_2$, y is sampled using the original sampler S.



(a) Description of $\mathbf{G}_0$. **(b)** Description of $\mathbf{G}_1$. **(c)** Description of $\mathbf{G}_2$.

Figure 16.4: Hybrids used in the proof of correctness of Lemma 16.1.

CLAIM 16.1. For all PPT adversaries $\mathcal{A}$, for all $m \in L$, there exists a PPT adversary $\mathcal{B}_2$, such that $|\Pr[out_2 = 1] - \Pr[out_1 = 1]| \leqslant \mathrm{Adv}^{\mathsf{close}}_{(S,\overline{S}^{-1}),\mathcal{B}_2,m}(\lambda)$.

Proof. Construct an adversary $\mathcal{B}_2$ breaking closeness. On input of (m,y), $\mathcal{B}_2$ computes $\bar{r} \leftarrow \overline{S}^{-1}(m,y)$, calls $\mathcal{A}$ on input of $(m,\bar{r},y)$ and outputs the resulting output. If y is sampled using $\overline{S}(m;r)$ (for $r \leftarrow \{0,1\}^{n(\lambda)}$), $\mathcal{B}_2$ perfectly simulates game $\mathbf{G}_1$ for $\mathcal{A}$. If y is sampled using $S(m;r)$ (for $r \leftarrow \{0,1\}^{p(\lambda)}$), $\mathcal{B}_2$ perfectly simulates game $\mathbf{G}_2$ for $\mathcal{A}$. Therefore,

$$\Pr\left[out_1 = 1\right] = \Pr\left[\mathrm{Exp}^{(1)\text{-close}}_{(S,\overline{S}^{-1}),\mathcal{B}_2,m}(\lambda) = 1\right] \text{ and}$$
$$\Pr\left[out_2 = 1\right] = \Pr\left[\mathrm{Exp}^{(0)\text{-close}}_{(S,\overline{S}^{-1}),\mathcal{B}_2,m}(\lambda) = 1\right].$$

□

Thus, we have that

$$\left|\Pr\left[out_2 = 1\right] - \Pr\left[out_0 = 1\right]\right| \leqslant \mathrm{Adv}^{\mathsf{close}}_{(S,\overline{S}^{-1}),\overline{\mathcal{B}},m}(\lambda) + \mathrm{Adv}^{\mathsf{inv}}_{(S,\overline{S}^{-1}),\overline{\mathcal{B}}',m}(\lambda)$$

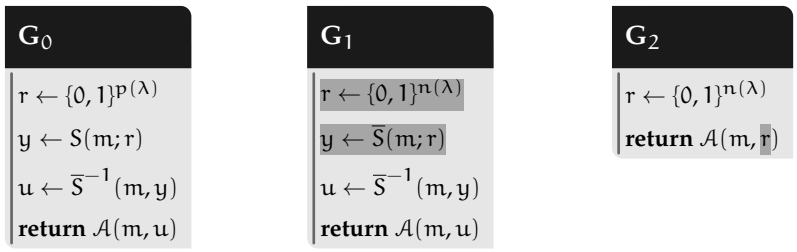
for some PPT adversaries $\overline{\mathcal{B}}$ and $\overline{\mathcal{B}}'$. Consider the adversary $\mathcal{A}$ distinguishing between game $\mathbf{G}_0$ and game $\mathbf{G}_2$ who on input of (m,r,y), outputs 0 if $\overline{S}(m;r) = y$ and outputs 1 otherwise. By definition, $\mathcal{A}$ always outputs 0 in $\mathbf{G}_0$. Hence,

$$\begin{aligned}\epsilon^{\mathsf{dec\text{-}error}}_{(E_S,D_S),m}(\lambda) &= \Pr\left[y \leftarrow S(m) : \overline{S}(m,\overline{S}^{-1}(m,y)) \neq y\right] \\ &= \Pr\left[out_{2,\mathcal{A}} = 1\right] \\ &= \left|\Pr\left[out_{2,\mathcal{A}} = 1\right] - \Pr\left[out_{0,\mathcal{A}} = 1\right]\right| \\ &\leqslant \mathrm{Adv}^{\mathsf{close}}_{(S,\overline{S}^{-1}),\overline{\mathcal{B}},m}(\lambda) + \mathrm{Adv}^{\mathsf{inv}}_{(S,\overline{S}^{-1}),\overline{\mathcal{B}}',m}(\lambda).\end{aligned}$$

PSEUDORANDOMNESS. We consider a sequence of hybrids starting from $\mathrm{Exp}^{(0)\text{-pre}}_{(E_S,D_S),\mathcal{A},m}(\lambda)$ and concluding with $\mathrm{Exp}^{(1)\text{-pre}}_{(E_S,D_S),\mathcal{A},m}(\lambda)$, see Figure 16.5.

GAME $\mathbf{G}_0$. $\mathbf{G}_0$ is identical to $\mathrm{Exp}^{(0)\text{-pre}}_{(E_S,D_S),\mathcal{A},m}(\lambda)$.

GAME $\mathbf{G}_1$. $\mathbf{G}_1$ is identical to $\mathbf{G}_0$ except that in $\mathbf{G}_0$, y is sampled using the original sampler S whereas in $\mathbf{G}_1$, y is sampled using the alternative sampler $\overline{S}$.



(a) Description of $\mathbf{G}_0$. **(b)** Description of $\mathbf{G}_1$. **(c)** Description of $\mathbf{G}_2$.

Figure 16.5: Hybrids used in the proof of pseudorandomness of Lemma 16.1.

CLAIM 16.2. For all PPT adversaries $\mathcal{A}$, for all $m \in L$, there exists a PPT adversary $\mathcal{B}_1$, such that $|\Pr[out_1 = 1] - \Pr[out_0 = 1]| \leqslant \mathrm{Adv}^{\mathsf{close}}_{(\overline{S},\overline{S}^{-1}),\mathcal{B}_1,m}(\lambda)$.

Proof. Construct a PPT adversary $\mathcal{B}_1$ on the closeness property as follows. On input of (m,y), $\mathcal{B}_1$ calls $\mathcal{A}$ on input of $(m,\overline{S}^{-1}(m,y))$ and outputs the resulting output. If $\mathcal{B}_1$ plays $\mathrm{Exp}^{(0)\text{-}\mathsf{close}}_{(\overline{S},\overline{S}^{-1}),\mathcal{B}_1,m}(\lambda)$, he simulates game $\mathbf{G}_0$. Otherwise, if $\mathcal{B}_1$ plays $\mathrm{Exp}^{(1)\text{-}\mathsf{close}}_{(\overline{S},\overline{S}^{-1}),\mathcal{B}_1,m}(\lambda)$, he simulates game $\mathbf{G}_1$. Hence,

$$\Pr\left[out_0 = 1\right] = \Pr\left[\mathrm{Exp}^{(0)\text{-}\mathsf{close}}_{(\overline{S},\overline{S}^{-1}),\mathcal{B}_1,m}(\lambda) = 1\right] \text{ and}$$
$$\Pr\left[out_1 = 1\right] = \Pr\left[\mathrm{Exp}^{(1)\text{-}\mathsf{close}}_{(\overline{S},\overline{S}^{-1}),\mathcal{B}_1,m}(\lambda) = 1\right].$$

□

GAME $\mathbf{G}_2$. $\mathbf{G}_2$ is identical to $\mathbf{G}_1$ except for the following difference. In $\mathbf{G}_1$, u is produced as inverse sampled random coins for y, whereas in $\mathbf{G}_2$ u, is a uniformly random bitstring from $\{0,1\}^{n(\lambda)}$. Hence, $\mathbf{G}_2$ is distributed identically to $\mathrm{Exp}^{(1)\text{-}\mathsf{pre}}_{(S,D_S),\mathcal{A},m}(\lambda)$.

CLAIM 16.3. For all PPT adversaries $\mathcal{A}$, for all $m \in L$, there exists a PPT adversary $\mathcal{B}_2$, such that $|\Pr[out_2 = 1] - \Pr[out_1 = 1]| \leqslant \mathrm{Adv}^{\mathsf{inv}}_{(\overline{S},\overline{S}^{-1}),\mathcal{B}_2,m}(\lambda)$.

Proof. We construct a PPT adversary $\mathcal{B}_2$ on the invertibility property. On input of (m,r,y), $\mathcal{B}_2$ calls $\mathcal{A}$ on input of (m,r) and outputs $\mathcal{A}$'s output. If $\mathcal{B}_2$ plays the game $\mathrm{Exp}^{(0)\text{-}\mathsf{inv}}_{(\overline{S},\overline{S}^{-1}),\mathcal{B}_2,m}(\lambda)$, he perfectly simulates $\mathbf{G}_1$. If $\mathcal{B}_2$ plays $\mathrm{Exp}^{(1)\text{-}\mathsf{inv}}_{(\overline{S},\overline{S}^{-1}),\mathcal{B}_2,m}(\lambda)$, he perfectly simulates $\mathbf{G}_2$. Therefore,

$$\Pr\left[out_1 = 1\right] = \Pr\left[\mathrm{Exp}^{(0)\text{-}\mathsf{inv}}_{(\overline{S},\overline{S}^{-1}),\mathcal{B}_2,m}(\lambda) = 1\right] \text{ and}$$
$$\Pr\left[out_2 = 1\right] = \Pr\left[\mathrm{Exp}^{(1)\text{-}\mathsf{inv}}_{(\overline{S},\overline{S}^{-1}),\mathcal{B}_2,m}(\lambda) = 1\right].$$

□

Hence,

$$\begin{aligned}\mathrm{Adv}^{\mathsf{pre}}_{(S,D_S),\mathcal{A},m}(\lambda) &= \left|\Pr\left[out_2 = 1\right] - \Pr\left[out_0 = 1\right]\right| \\ &\leqslant \mathrm{Adv}^{\mathsf{close}}_{(\overline{S},\overline{S}^{-1}),\overline{\mathcal{B}},m}(\lambda) + \mathrm{Adv}^{\mathsf{inv}}_{(\overline{S},\overline{S}^{-1}),\overline{\mathcal{B}}',m}(\lambda)\end{aligned}$$

for some PPT adversaries $\overline{\mathcal{B}}$ and $\overline{\mathcal{B}}'$. □

16.3.2 Every Pseudorandomly Encodable Distribution can be Inverse Sampled

LEMMA 16.2. *Let* $\alpha \in \{\approx_c, \equiv_s\}$ *and* $\beta \in \{\mathsf{rand}, \mathsf{det}\}$. *If* $\mathsf{PREH}^{\beta}_{\alpha}$ *holds, then* $\mathsf{ISH}^{\beta}_{\alpha}$ *holds.*

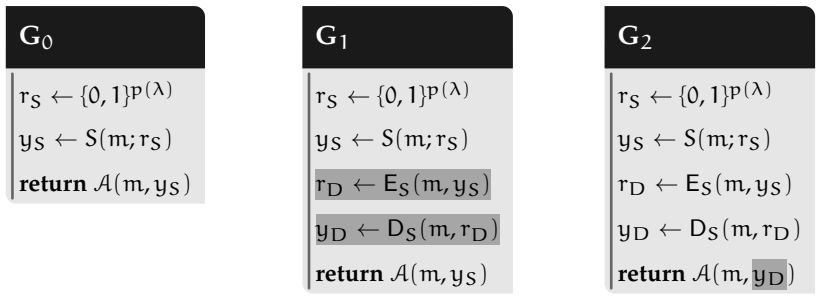
Proof. We prove the statement for the computational randomized case. The remaining cases are similar.

Assume $\mathsf{PREH}^{\mathsf{rand}}_{\approx_c}$ holds. Let S be a PPT algorithm. $\mathsf{PREH}^{\mathsf{rand}}_{\approx_c}$ implies that for S there exist efficient algorithms E_S (possibly randomized) with output length $n(\lambda)$ and D_S (deterministic) satisfying correctness and pseudorandomness.

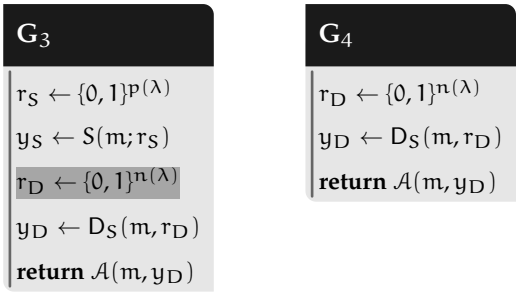
For $m \in L$, $r \in \{0,1\}^{n(\lambda)}$, $y \in \{0,1\}^*$, we define the alternative sampler and the inverse sampler as follows:

$$\overline{S}(m; r) := D_S(m, r),$$
$$\overline{S}^{-1}(m, y) := E_S(m, y).$$

CLOSENESS. Let $\mathcal{A}$ be an adversary breaking closeness. We consider a sequence of games starting from the game $\mathsf{Exp}^{(0)\text{-close}}_{(\overline{S},\overline{S}^{-1}),\mathcal{A},m}(\lambda)$ and concluding with $\mathsf{Exp}^{(1)\text{-close}}_{(\overline{S},\overline{S}^{-1}),\mathcal{A},m}(\lambda)$, see Figure 16.6.



(a) Description of $\mathbf{G}_0$. **(b)** Description of $\mathbf{G}_1$. **(c)** Description of $\mathbf{G}_2$.



(d) Description of $\mathbf{G}_3$. **(e)** Description of $\mathbf{G}_4$.

Figure 16.6: Hybrids used in the proof of closeness of Lemma 16.2.

GAME $\mathbf{G}_0$. $\mathbf{G}_0$ is identical to $\mathsf{Exp}^{(0)\text{-close}}_{(\overline{S},\overline{S}^{-1}),\mathcal{A},m}(\lambda)$.

GAME $\mathbf{G}_1$. $\mathbf{G}_1$ is the same as $\mathbf{G}_0$ except for the conceptional difference that $\mathbf{G}_1$ additionally samples $r_D \leftarrow E_S(m, y_S)$ and $y_D \leftarrow D_S(m, r_D)$. This difference is only conceptional since the view of the adversary is not affected. Hence, $\Pr[out_0 = 1] = \Pr[out_1 = 1]$.

GAME $\mathbf{G}_2$. $\mathbf{G}_2$ is identical to $\mathbf{G}_1$ except that $\mathbf{G}_2$ passes the value y_D to $\mathcal{A}$ instead of the value y_S.

Hence, $\mathbf{G}_1$ and $\mathbf{G}_2$ proceed exactly identical if $y_S = y_D$. More formally, let F be the event that $y_S \neq y_D$. Then, $out_1 = 1 \wedge \neg F \Leftrightarrow out_2 = 1 \wedge \neg F$. Hence, the Difference Lemma (due to Shoup [Sho04]) bounds

$$\left|\Pr\left[out_2 = 1\right] - \Pr\left[out_1 = 1\right]\right| \leqslant \Pr[F].$$

Furthermore, correctness guarantees that for all $m \in L$,

$$\Pr[F] = \Pr\left[y_S \leftarrow S(m) \colon D_S(m, E_S(m, y_S)) \neq y_S\right] = \epsilon^{\text{dec-error}}_{(E_S,D_S),m}(\lambda)$$

is negligible.

GAME $\mathbf{G}_3$. $\mathbf{G}_3$ is identical to $\mathbf{G}_2$ except for one difference. In $\mathbf{G}_2$, r_D is produced as $E_S(m, y_S)$, whereas in $\mathbf{G}_3$, r_D is sampled uniformly from $\{0,1\}^{n(\lambda)}$.

CLAIM 16.4. For all PPT adversaries $\mathcal{A}$, for all $m \in L$, there exists a PPT adversary $\mathcal{B}_3$, such that $|\Pr[out_3 = 1] - \Pr[out_2 = 1]| \leqslant \mathrm{Adv}^{\text{pre}}_{(E_S,D_S),\mathcal{B}_3,m}(\lambda)$.

Proof. Construct an adversary $\mathcal{B}_3$ breaking pseudorandomness as follows. On input of $(m, u =: r_D)$, $\mathcal{B}_3$ calls $\mathcal{A}$ on input $(m, D_S(m, r_D))$ and outputs the resulting output. If u is produced via $E_S(m, y_S)$ for $y_S \leftarrow S(m)$, $\mathcal{B}_3$ perfectly simulates game $\mathbf{G}_2$. Otherwise, if u is uniformly random over $\{0,1\}^{n(\lambda)}$, $\mathcal{B}_3$ perfectly simulates game $\mathbf{G}_3$. Hence,

$$\Pr\left[out_3 = 1\right] = \Pr\left[\mathrm{Exp}^{(1)\text{-pre}}_{(E_S,D_S),\mathcal{B}_3,m}(\lambda) = 1\right] \text{ and}$$
$$\Pr\left[out_2 = 1\right] = \Pr\left[\mathrm{Exp}^{(0)\text{-pre}}_{(E_S,D_S),\mathcal{B}_3,m}(\lambda) = 1\right].$$

□

GAME $\mathbf{G}_4$. The game $\mathbf{G}_4$ is identical to $\mathbf{G}_3$ except for the conceptional difference that $\mathbf{G}_4$ does not sample r_D and y_D since these values are never used. Hence, $\Pr[out_3 = 1] = \Pr[out_4 = 1]$. $\mathbf{G}_4$ is identical to $\mathrm{Exp}^{(1)\text{-close}}_{(S,S^{-1}),\mathcal{A},m}(\lambda)$.

Hence,

$$\begin{aligned}\mathrm{Adv}^{\text{close}}_{(S,S^{-1}),\mathcal{A},m}(\lambda) &= \left|\Pr\left[out_4 = 1\right] - \Pr\left[out_0 = 1\right]\right| \\ &\leqslant \mathrm{Adv}^{\text{pre}}_{(E_S,D_S),\overline{\mathcal{B}},m}(\lambda) + \epsilon^{\text{dec-error}}_{(E_S,D_S),m}(\lambda)\end{aligned}$$

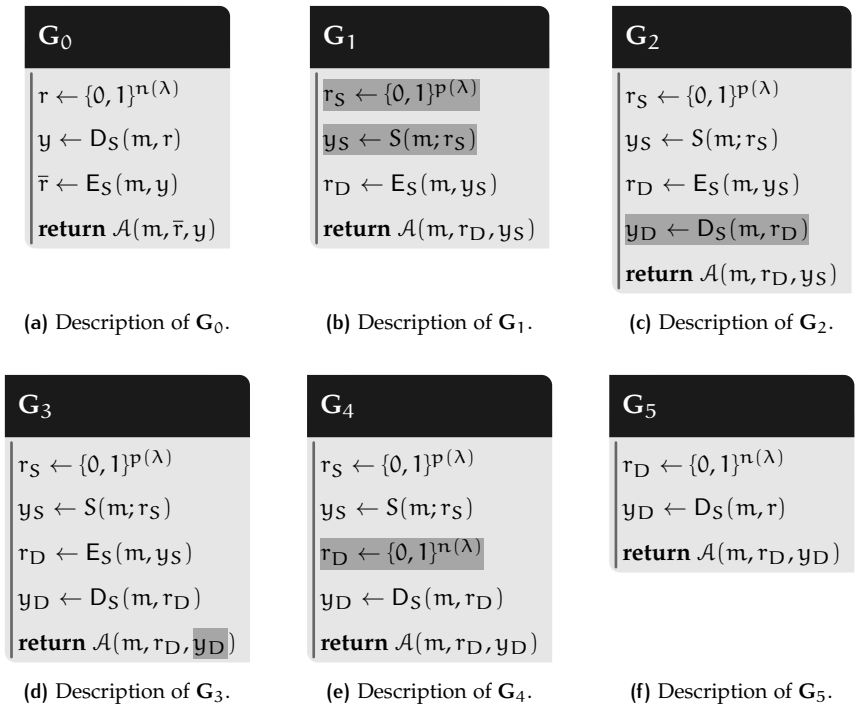
for some PPT adversary $\overline{\mathcal{B}}$.

INVERTIBILITY. We consider a sequence of hybrids starting from the game $\mathrm{Exp}^{(1)\text{-inv}}_{(S,S^{-1}),\mathcal{A},m}(\lambda)$ and concluding with $\mathrm{Exp}^{(0)\text{-inv}}_{(S,S^{-1}),\mathcal{A},m}(\lambda)$, see Figure 16.7.

GAME $\mathbf{G}_0$. $\mathbf{G}_0$ is identical to $\mathrm{Exp}^{(1)\text{-inv}}_{(S,S^{-1}),\mathcal{A},m}(\lambda)$.

GAME $\mathbf{G}_1$. $\mathbf{G}_1$ is identical to $\mathbf{G}_0$ up to a small difference. In $\mathbf{G}_0$, y is sampled using D_S on uniform randomness. In $\mathbf{G}_1$, y is sampled using the original sampler S.

CLAIM 16.5. For all PPT adversaries $\mathcal{A}$, for all $m \in L$, there exists a PPT adversary $\mathcal{B}_1$, such that $|\Pr[out_1 = 1] - \Pr[out_0 = 1]| \leqslant \mathrm{Adv}^{\text{pre}}_{(E_S,D_S),\mathcal{B}_1,m}(\lambda) + \epsilon^{\text{dec-error}}_{(E_S,D_S),m}(\lambda)$.



(a) Description of $\mathbf{G}_0$. **(b)** Description of $\mathbf{G}_1$. **(c)** Description of $\mathbf{G}_2$.

(d) Description of $\mathbf{G}_3$. **(e)** Description of $\mathbf{G}_4$. **(f)** Description of $\mathbf{G}_5$.

Figure 16.7: Hybrids used in the proof of invertibility of Lemma 16.2.

Proof. Let $\mathcal{A}$ be an adversary distinguishing $\mathbf{G}_0$ and $\mathbf{G}_1$. Construct an adversary $\mathcal{B}_1$ on the closeness property. On input of (m, y), $\mathcal{B}_1$ computes $\bar{r} \leftarrow \mathsf{E}_S(m, y)$ and calls $\mathcal{A}$ on input $(m, \bar{r}, y)$. If y is sampled via $\overline{S}(m)$, $\mathcal{B}_1$ simulates game $\mathbf{G}_0$ for $\mathcal{A}$. Else, if y is sampled via $S(m)$, $\mathcal{B}_1$ simulates game $\mathbf{G}_1$ for $\mathcal{A}$. Hence,

$$\left|\Pr\left[out_1 = 1\right] - \Pr\left[out_0 = 1\right]\right| \leqslant \mathrm{Adv}^{\mathsf{close}}_{(\overline{S},\overline{S}^{-1}),\mathcal{B}_1,m}(\lambda).$$

□

GAME $\mathbf{G}_2$. $\mathbf{G}_2$ is identical to $\mathbf{G}_1$ except for the conceptional difference that $\mathbf{G}_2$ additionally computes y_D as $\mathsf{D}_S(m, r_D)$ but never uses this value. Hence, $\Pr[out_1 = 1] = \Pr[out_2 = 1]$.

GAME $\mathbf{G}_3$. $\mathbf{G}_3$ is identical to $\mathbf{G}_2$ except that $\mathbf{G}_3$ passes the value y_D to $\mathcal{A}$ instead of the value y_S. $\mathbf{G}_2$ and $\mathbf{G}_3$ behave identical if $y_D = y_S$. Let F denote the failure event $y_D \neq y_S$. Hence, $out_2 = 1 \wedge \neg \Leftrightarrow out_3 = 1 \wedge \neg\mathsf{F}$. The Difference Lemma (due to Shoup [Sho04]) bounds

$$\left|\Pr\left[out_3 = 1\right] - \Pr\left[out_2 = 1\right]\right| \leqslant \Pr[\mathsf{F}].$$

Due to correctness, for all $m \in L$,

$$\Pr[\mathsf{F}] = \Pr\left[y_S \leftarrow S(m) \colon \mathsf{D}_S(m, \mathsf{E}_S(m, y_S)) \neq y_S\right] = \epsilon^{\mathsf{dec\text{-}error}}_{(\mathsf{E}_S,\mathsf{D}_S),m}(\lambda)$$

is negligible.

GAME $\mathbf{G}_4$. $\mathbf{G}_4$ is identical to $\mathbf{G}_3$ except that $\mathbf{G}_3$ produces r_D via $\mathsf{E}_S(m, y_S)$, whereas $\mathbf{G}_4$ samples r_D uniformly from $\{0,1\}^{n(\lambda)}$.

CLAIM 16.6. For all PPT adversaries $\mathcal{A}$, for all $m \in L$, there exists a PPT adversary $\mathcal{B}_4$, such that $|\Pr[out_4 = 1] - \Pr[out_3 = 1]| \leqslant \mathrm{Adv}^{\mathsf{pre}}_{(\mathsf{E}_S,\mathsf{D}_S),\mathcal{B}_4,m}(\lambda)$.

Proof. Construct a PPT adversary $\mathcal{B}_4$ breaking the pseudorandomness property. On input of (m, u), $\mathcal{B}_4$ calls $\mathcal{A}$ on input of $(m, u =: r_D, \mathsf{D}_S(m, u) =: y_D)$ and outputs the resulting output. If u is sampled via $\mathsf{E}_S(m, y)$ for $y \leftarrow S(m)$, $\mathcal{B}_4$ perfectly simulates game $\mathbf{G}_3$ for $\mathcal{A}$. Otherwise, if u is uniformly random over $\{0,1\}^{n(\lambda)}$, $\mathcal{B}_4$ perfectly simulates game $\mathbf{G}_4$ for $\mathcal{A}$. Hence,

$$\Pr\left[out_3 = 1\right] = \Pr\left[\mathrm{Exp}^{(0)\text{-}\mathsf{pre}}_{(\mathsf{E}_S,\mathsf{D}_S),\mathcal{B}_4,m}(\lambda) = 1\right] \text{ and}$$
$$\Pr\left[out_4 = 1\right] = \Pr\left[\mathrm{Exp}^{(1)\text{-}\mathsf{pre}}_{(\mathsf{E}_S,\mathsf{D}_S),\mathcal{B}_4,m}(\lambda) = 1\right].$$

□

GAME $\mathbf{G}_5$. The difference between $\mathbf{G}_4$ and $\mathbf{G}_5$ is that $\mathbf{G}_5$ does not produce the obsolete values r_S and y_S. Hence, this game hop is again only conceptional and $\Pr[out_4 = 1] = \Pr[out_5 = 1]$.

Hence,

$$\left|\Pr\left[out_5 = 1\right] - \Pr\left[out_0 = 1\right]\right| \leqslant 2 \cdot \mathrm{Adv}^{\mathsf{pre}}_{(\mathsf{E}_S,\mathsf{D}_S),\overline{\mathcal{B}},m}(\lambda) + 2 \cdot \epsilon^{\mathsf{dec\text{-}error}}_{(\mathsf{E}_S,\mathsf{D}_S),m}(\lambda)$$

for some PPT adversary $\overline{\mathcal{B}}$. □

The above proof directly generalizes to the non-adaptive and adaptive variants of pseudorandom encodings and invertible sampling with public parameters. See Figure 16.8 for a proof sketch for the adaptive version.

Theorem 16.1 together with Theorem 15.1 yields the following corollaries.

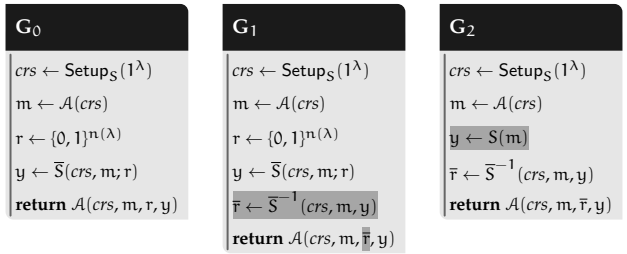
COROLLARY 16.1. *Let* $\alpha \in \{\approx_c, \equiv_s\}$ *and* $\beta \in \{\mathsf{rand}, \mathsf{det}\}$. *If* $\mathsf{cPREH}^{\beta}_{\alpha}$ *is true, then* $\mathsf{acISH}^{\beta}_{\alpha}$ *is true.*

COROLLARY 16.2. *Let* $\alpha \in \{\approx_c, \equiv_s\}$ *and* $\beta \in \{\mathsf{rand}, \mathsf{det}\}$. *If* $\mathsf{cISH}^{\beta}_{\alpha}$ *is true, then* $\mathsf{acISH}^{\beta}_{\alpha}$ *is true.*

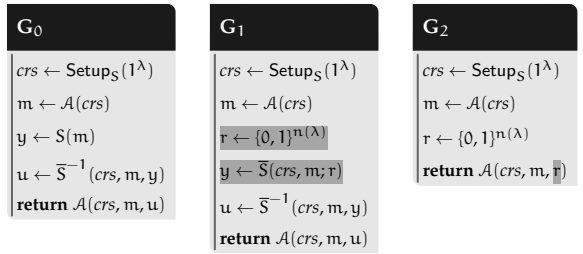
This is an excellent example of the potential pseudorandom encodings offer. Looking through the lens of invertible sampling, it is rather unclear how the static and adaptive notions relate, whereas pseudorandom encodings directly provide a static-to-adaptive transformation.

Particularly, Corollary 16.2 together with [DKR15] yields the first instantiation of an *adaptive* explainability compiler without complexity leveraging and, hence, based only on polynomial hardness assumptions. The recent paper [CsW19] uses such an adaptive explainability compiler to obtain adaptive MPC with communication complexity which is sublinear in the circuit size. Their construction relies on complexity leveraging which entails a subexponential loss relative to IO and one-way functions. Hence, we obtain the following corollary improving on Theorem 7 in the proceedings version of [CsW19] in a black-box way.

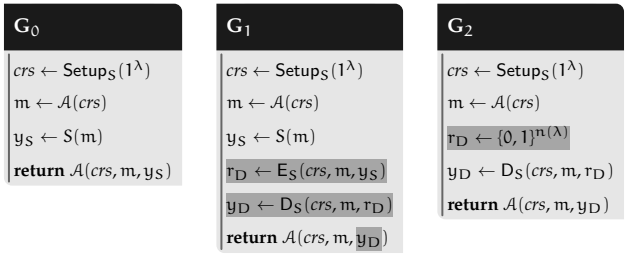
COROLLARY 16.3. *Assuming polynomially secure IO and the adaptive hardness of LWE, then succinct adaptive two-round MPC in the malicious setting is possible (for all deterministic* n*-party functionalities* $f\colon (\{0,1\}^{\ell_{in}})^n \to \{0,1\}^{\ell_{out}}$ *with circuit depth* d*) in the global CRS model such that the size of the CRS, the communication complexity, and online-computational complexity of the protocol are polynomial in* λ, ℓ_{in}, ℓ_{out}, d *and* n.



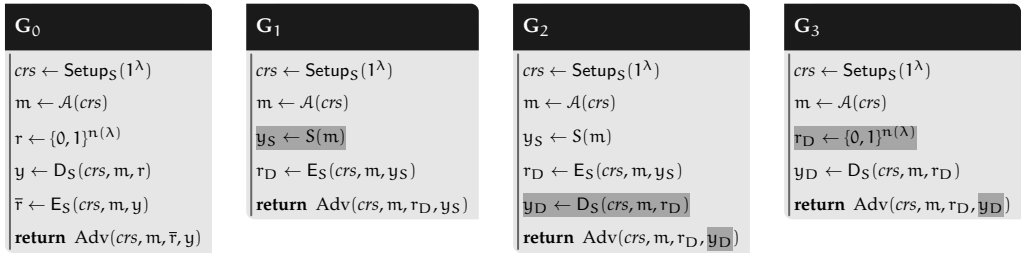
(a) *Adaptive correctness.* In $\mathbf{G}_0$, for adaptively chosen m, we have $D_S(crs, m, r) = \overline{S}(crs, m; r) = y$. Game hop from $\mathbf{G}_0$ to $\mathbf{G}_1$ follows by adaptive invertibility. Game hop from $\mathbf{G}_1$ to $\mathbf{G}_2$ follows from adaptive closeness. Hence, in $\mathbf{G}_2$, $D_S(crs, m, r) = \overline{S}(crs, m; r) = y$, where $r = E_S(crs, m, y) = \overline{S}^{-1}(crs, m, y)$.



(b) *Adaptive pseudorandomness.* $\mathbf{G}_0$ corresponds to adaptive pseudorandomness game, where adversary receives encoded randomness, that is $E_S(crs, m, y) = \overline{S}^{-1}(crs, m, y)$. Game hop from $\mathbf{G}_0$ to $\mathbf{G}_1$ is justified by adaptive closeness. Game hop from $\mathbf{G}_1$ to $\mathbf{G}_2$ is justified by adaptive invertibility. $\mathbf{G}_2$ corresponds to adaptive pseudorandomness game, where adversary receives true randomness.



(c) *Adaptive closeness.* $\mathbf{G}_0$ corresponds to the adaptive closeness game, where $\mathcal{A}$ receives a sample from S. If y_S and y_D are identical, $\mathbf{G}_0$ and $\mathbf{G}_1$ are identical. Adaptive correctness guarantees this with overwhelming probability. The game hop from $\mathbf{G}_1$ to $\mathbf{G}_2$ is justified by adaptive pseudorandomness. $\mathbf{G}_2$ corresponds to the adaptive closeness game, where $\mathcal{A}$ receives a sample from $\overline{S}$.



(d) *Adaptive invertibility.* Game $\mathbf{G}_0$ corresponds to the adaptive invertibility game, where $\mathcal{A}$ receives inverse sampled random coins. The game hop from $\mathbf{G}_0$ to $\mathbf{G}_1$ is justified by adaptive closeness which follows from adaptive correctness and adaptive pseudorandomness. The game hop from $\mathbf{G}_1$ to $\mathbf{G}_2$ is justified by adaptive correctness and the game hop from $\mathbf{G}_2$ to $\mathbf{G}_3$ is justified by adaptive pseudorandomness. $\mathbf{G}_3$ corresponds to the adaptive invertibility game, where $\mathcal{A}$ receives the actual randomness.

Figure 16.8: Proof sketch for the equivalence of the adaptive notions of pseudorandom encodings and invertible sampling.

17 A TAXONOMY OF PSEUDORANDOM ENCODINGS

In this chapter we classify the different variants of the pseudorandom encoding hypothesis. In Section 17.1, we study the pseudorandom encoding hypothesis with deterministic encoding algorithm and identify a relation to compression. In Section 17.2, we study the pseudorandom encoding hypothesis with randomized encoding and its conflicts with extractable one-way functions. In Section 17.3, we describe an instantiation of $\mathsf{cPREH}^{\mathsf{rand}}_{\approx_c}$ (with and without universal setup) based on indistinguishability obfuscation and one-way functions due to [SW14; DKR15]. Finally, in Section 17.4, we bootstrap $\mathsf{cPREH}^{\mathsf{rand}}_{\approx_c}$ with a common *random* string from the construction in Section 17.3 in conjunction with *weak* $\mathsf{cPREH}^{\mathsf{rand}}_{\approx_c}$ with a common *random* string for the setup algorithm from Section 17.3.

17.1 DETERMINISTIC ENCODING ALGORITHM

For the purpose of classifying pseudorandom encodings with a deterministic encoding algorithm, we first introduce some notions of entropy and computational analogues thereof.

Min-entropy captures the ability to guess the value of a distribution in a single attempt, cf. Definition 2.2. However, this is a very pessimistic view. For many purposes, it suffices to work with a distribution which is statistically close (i. e., has negligible statistical distance) to a distribution with high min-entropy.[61]

[61] *We note that if some distribution X is ϵ-close to a distribution Y, $H_\infty(X) \geqslant -\log(2^{-H_\infty(Y)} + \epsilon(\lambda)) = H_\infty(Y) - \log(1 + \epsilon(\lambda)2^{H_\infty(Y)})$.*

DEFINITION 17.1 (ϵ-smooth min-entropy, [Rey11]). A source X has *ϵ-smooth min-entropy* at least k, denoted as $H^{\epsilon}_{\infty}(X) \geqslant k$, if there exists a distribution X' with $\Delta(X, X') \leqslant \epsilon$ such that $H_\infty(X') \geqslant k$.

In many cases, some information Z that is correlated to the actual source X is known. Since for our purposes, the conditional part Z is not under adversarial control, we use the notion of *average* conditional min-entropy as used in [HLR07; DORS08], cf. Definition 2.3. That is, the average min-entropy of some distribution X conditioned on Z is defined as $-\log(\mathbb{E}_{z \leftarrow Z}[\max_x \Pr[X_z = x]])$.

The notion of average min-entropy can be relaxed in a similar way as Definition 17.1 as follows.

DEFINITION 17.2 (Average ϵ-smooth min-entropy, [DORS08; Rey11]). Let (X, Z) be a joint distribution. The distribution X has *average ϵ-smooth min-entropy* at least k *conditioned on* Z, denoted as $\widetilde{H}^{\epsilon}_{\infty}(X \mid Z) \geqslant k$, if there exists a joint distribution (X', Z') with $\Delta((X, Z), (X', Z')) \leqslant \epsilon$ such that $\widetilde{H}_\infty(X' \mid Z') \geqslant k$.

Let X be an efficiently samplable distribution. In the literature, there are several computational notions of entropy. HILL entropy [HILL99] constitutes a natural computational variant of min-entropy. A source has high HILL

entropy, if it is computationally indistinguishable from a source that has high min-entropy.

DEFINITION 17.3 (HILL entropy, [HILL99; BSW03]). A distribution X has *HILL entropy* at least k, denoted by $H^{\mathsf{HILL}}_{\epsilon,s}(X) \geqslant k$, if there exists a distribution Y such that $H_\infty(Y) \geqslant k$ and $|\Pr[x \leftarrow X\colon \mathcal{A}(x) = 1] - \Pr[y \leftarrow Y\colon \mathcal{A}(y) = 1]| \leqslant \epsilon$ for all $\mathcal{A}$ of size at most s.

Shannon's source coding theorem [Sha48] states that the minimum compression length (over all compression and decompression algorithms) of a distribution equals its average entropy (up to small additive terms). Yao entropy [Yao82] constitutes the corresponding computational counterpart. Intuitively, a source has high Yao entropy, if it cannot be *efficiently* compressed.

DEFINITION 17.4 (Yao entropy, [Yao82; BSW03]). A distribution X has *Yao entropy* at least k, denoted by $H^{\mathsf{Yao}}_{\epsilon,s}(X) \geqslant k$, if for every pair of circuits (E, D) of total size s with outputs of E having length ℓ, $\Pr_{x \leftarrow X}[D(E(x)) = x] \leqslant 2^{\ell-k} + \epsilon$.

When we omit the subscripts for H^{HILL} and H^{Yao}, we mean $H^{\mathsf{HILL}}_{\epsilon,s}$ and $H^{\mathsf{Yao}}_{\epsilon,s}$ for any negligible ϵ and polynomial s, respectively. Since compressibility implies distinguishability, HILL entropy implies Yao entropy. The converse, however, is believed to be false, [Wee04; HLR07].

DEFINITION 17.5 (Conditional HILL entropy, [HLR07]). For a distribution (X, Z), we say that X has HILL entropy at least k conditioned on Z, denoted by $H^{\mathsf{HILL}}_{\epsilon,s}(X \mid Z) \geqslant k$, if there exists a collection of distributions Y_z giving rise to a joint distribution (Y, Z), such that $\widetilde{H}_\infty(Y \mid Z) \geqslant k$ and $|\Pr[(x, z) \leftarrow (X, Z)\colon \mathcal{A}(x, z) = 1] - \Pr[(y, z) \leftarrow (Y, Z)\colon \mathcal{A}(y, z) = 1]| \leqslant \epsilon$ for all circuits $\mathcal{A}$ of size at most s.

Conditional Yao entropy is defined by simply giving the compressor and decompressor algorithm the value z as input.

DEFINITION 17.6 (Conditional Yao entropy, [HLR07]). For a distribution (X, Z), we say that X has Yao entropy at least k conditioned on Z, denoted by $H^{\mathsf{Yao}}_{\epsilon,s}(X \mid Z) \geqslant k$, if for every pair of circuits (E, D) of total size at most s with outputs of E having length ℓ, $\Pr_{(x,z) \leftarrow (X,Z)}[D(z, E(z, x)) = x] \leqslant 2^{\ell-k} + \epsilon$.

17.1.1 Information-theoretic Guarantees and Compression

Traditionally, the theory of compression mostly considers families of sources that are not indexed by strings. Let Σ be some alphabet. In this part, we always consider $\Sigma := \{0, 1\}$. Let X be a source.

DEFINITION 17.7 ([Wee04; TVZ05]). Let $E_X\colon \Sigma^* \to \Sigma^*$ and $D_X\colon \Sigma^* \to \Sigma^*$ be functions. We say (E_X, D_X) *compresses source* X *to length* m *with decoding error* ϵ if

- $\Pr\left[x \leftarrow X\colon D_X(E_X(x)) \neq x\right] \leqslant \epsilon$, and
- $\mathbb{E}\left[|E_X(X)|\right] \leqslant m$.

DEFINITION 17.8 ([Wee04; TVZ05]). We say source X is *compressible to length (exactly)* m if there exist functions E_X and D_X such that (E_X, D_X) compresses X to length (exactly) m.

The decoding error can be eliminated at the cost of longer codewords.

LEMMA 17.1 ([TVZ05]). *Let X_λ be a source on $\{0,1\}^\lambda$ which is compressible to length m with decoding error ϵ by algorithms (E_X, D_X). Further, let $m_0 \in \mathbb{N}$ be a lower bound on the output length of E_X, i.e., such that for all $x \in \mathrm{supp}(X_\lambda)$, $|E_X(x)| \geqslant m_0$. Then, X_λ is compressible to length $m + \epsilon(\lambda - m_0) + 1$ with decoding error 0.*

Proof sketch. To show this, [TVZ05] construct an encoding algorithm E'_X which on input of x tests if $D_X(E_X(x)) = x$. If this is the case, E'_X outputs $0 \,\|\, E_X(x)$. Else, E'_X outputs $1 \,\|\, x$. □

The statistical deterministic variant of the pseudorandom encoding hypothesis is strongly related to compression.

THEOREM 17.1. *If (weak) $\mathsf{PREH}^{\mathsf{det}}_{\equiv_s}$ is true for X, i.e., there exist deterministic algorithms (E_X, D_X) with E_X having output length n satisfying correctness and pseudorandomness. Then, the ϵ-smooth min-entropy $H^\epsilon_\infty(X) \geqslant n$ for some negligible function ϵ.*

Proof. Consider the distribution $Y' \coloneqq D_X(U_n)$. As already seen in the proof of Theorem 16.1, Y' and X are statistically indistinguishable due to correctness and pseudorandomness. Hence, the statistical distance $\Delta(X, Y') \leqslant \delta(\lambda)$ for some negligible δ.

Due to correctness and since E_X is deterministic,

$$\begin{aligned}\Pr\left[x \leftarrow X\colon D_X(E_X(x)) = x\right] &= \Pr\left[x \leftarrow X\colon E_X(D_X(E_X(x))) = E_X(x)\right] \\ &\geqslant 1 - \nu(\lambda)\end{aligned}$$

for some negligible function ν. Applying pseudorandomness, we get that the probability

$$\Pr\left[u \leftarrow U_n\colon E_X(D_X(u)) = u\right] \geqslant 1 - \nu'(\lambda)$$

for some negligible function ν', where the probability is only over the choice of u. Therefore, D_X operates almost injectively on the set $\{0,1\}^n$. More formally, let $V_0 \subset \{0,1\}^n$ denote the set of all u such that $E_X(D_X(u)) = u$ and let $V_1 \coloneqq \{0,1\}^n \setminus V_0$. We have that, $|V_1|/|\{0,1\}^n| \leqslant \nu'(\lambda)$. Let $\overline{V_1}$ be some arbitrary subset of $\{0,1\}^{k(\lambda)} \setminus D_X(V_0)$ such that $|\overline{V_1}| = |V_1|$ (for some polynomial k). Let Y'' be the uniform distribution over $D_X(V_0) \cup \overline{V_1}$. Note that $|D_X(V_0) \cup \overline{V_1}| = 2^n$ and $\Pr[Y' \in \overline{V_1}] \leqslant \Pr[Y' \notin D_X(V_0)] \leqslant \nu'(\lambda)$. The statistical distance between Y' and Y'' is

$$\begin{aligned}\Delta(Y', Y'') &= \sum_{a \in D_X(V_0) \cup \overline{V_1}} \left|\Pr[Y' = a] - \Pr[Y'' = a]\right| \\ &= \underbrace{\sum_{a \in D_X(V_0)} \left|\Pr[Y' = a] - \Pr[Y'' = a]\right|}_{\leqslant 1 - |V_0| \cdot 2^{-n} \leqslant 1 - (1 - \nu'(\lambda))} \\ &\quad + \underbrace{\sum_{a \in \overline{V_1}} \left|\Pr[Y' = a] - \Pr[Y'' = a]\right|}_{\leqslant \sum_{a \in \overline{V_1}} (\Pr[Y' = a] + \Pr[Y'' = a]) \leqslant 2 \cdot \nu'(\lambda)} \leqslant 3 \cdot \nu'(\lambda)\end{aligned}$$

and hence negligible. Furthermore, since Y'' is the uniform distribution over a set of size 2^n, its min-entropy equals n. Since $\Delta(X, Y') \leqslant \delta(\lambda)$ and $\Delta(Y', Y'') \leqslant 3\nu'(\lambda)$ for negligible functions δ and ν', we have

$$H_\infty^{\delta+3\nu'}(X) \geqslant H_\infty(Y'') = n.$$

□

Theorem 17.1 directly yields the following corollary.

COROLLARY 17.1. *If* $\mathsf{PREH}^{\mathsf{det}}_{\equiv_s}$ *is true for some source* X*, there exists an* $n \leqslant H^\epsilon_\infty(X)$ *(for some negligible function* ϵ*), such that* X *is compressible to length exactly* n*. The decoding error can then be eliminated applying Lemma 17.1.*

Hence, for $\mathsf{PREH}^{\mathsf{det}}_{\equiv_s}$ to be true for a source X, it is necessary that X is efficiently compressible. However, this is not a sufficient criterion since a compression algorithm can have some structure which makes it easily distinguishable from uniform randomness (e. g., the compression algorithm used in the proof of Lemma 17.1).

The distribution induced by pseudorandom generators can provably not be compressed. However, those distributions have low ϵ-smooth min-entropy.

LEMMA 17.2. *Let* iPRG *be an injective pseudorandom generator with polynomial stretch* $\mathrm{poly}(\cdot)$ *and let* ϵ *be a negligible function. Then,* $H^\epsilon_\infty(\mathsf{iPRG}(U_\lambda)) \leqslant \lambda + \delta(\lambda)$ *for some negligible function* δ*.*

Proof. Since $\mathsf{iPRG}(U_\lambda)$ is a uniform distribution over $\mathsf{iPRG}(\{0,1\}^\lambda)$, it suffices to consider all ϵ-close flat distributions over $\{0,1\}^{\mathrm{poly}(\lambda)}$ to obtain an upper bound on $H^\epsilon_\infty(\mathsf{iPRG}(U_\lambda))$. Let D be the uniform distribution over $\mathsf{iPRG}(\{0,1\}^\lambda) \cup A$, for some set A such that $\mathsf{iPRG}(\{0,1\}^\lambda) \cap A = \varnothing$. Let $|A| \coloneqq k$. Then,

$$\begin{aligned}\Delta(\mathsf{iPRG}(U_\lambda), D) &= \sum_{a \in \mathsf{iPRG}(\{0,1\}^\lambda)} \left|\frac{1}{2^\lambda} - \frac{1}{2^\lambda + k}\right| + \sum_{a \in A} \left|\frac{1}{2^\lambda + k}\right| \\ &= 1 - \frac{2^\lambda}{2^\lambda + k} + \frac{k}{2^\lambda + k} = \frac{2k}{2^\lambda + k}.\end{aligned}$$

Hence, for $\Delta(\mathsf{iPRG}(U_\lambda), D) \leqslant \epsilon(\lambda)$, k is upper bounded by

$$k \leqslant k\underbrace{(2 - \epsilon(\lambda))}_{\geqslant 1} \leqslant 2^\lambda \cdot \epsilon(\lambda).$$

Therefore, the min-entropy of D is at most $H_\infty(D) = \log(2^\lambda + k) \leqslant \log(2^\lambda + 2^\lambda\epsilon) = \lambda + \log(1+\epsilon)$, which is negligibly close to λ. Hence, $H^\epsilon_\infty(\mathsf{iPRG}(U_\lambda)) \leqslant \lambda + \delta(\lambda)$ for some negligible function δ. □

Thus, if injective PRGs exist (and since the distribution induced by iPRG is clearly compatible, cf. Definition 15.3), then $\mathsf{PREH}^{\mathsf{det}}_{\equiv_s}$ is false. We can further strengthen this result by considering general PRGs.

LEMMA 17.3. *Let* PRG *be a pseudorandom generator with polynomial stretch* $\mathrm{poly}(\cdot)$ *and let* ϵ *be a negligible function. Then,* $H^\epsilon_\infty(\mathsf{PRG}(U_\lambda)) \leqslant \lambda + \delta(\lambda)$ *for some negligible function* δ*.*

Proof. Due to Lemma 15.1, there exists a uniform distribution D over a set of size 2^λ such that $\mathsf{PRG}(U_\lambda)$ and D are statistically close. By a similar argument as in Lemma 17.2, in order to upper bound the ϵ-smooth min-entropy of $\mathsf{PRG}(U_\lambda)$, it suffices to consider all ϵ-close flat distributions. Then, by the same computations as in the proof of Lemma 17.2, Lemma 17.3 follows. □

Theorem 17.1 together with Lemmas 15.1 and 17.3 yield the following corollary.

COROLLARY 17.2. *If one-way functions exist,* $\mathsf{PREH}^{\mathsf{det}}_{\equiv_s}$ *is false.*

INFORMATION-THEORETIC GUARANTEES WITH SETUP. Intuitively, a common reference string cannot add additional entropy to $\mathsf{E}_X(crs, X)$. That is, if $\mathsf{cPREH}^{\mathsf{det}}_{\equiv_s}$ is true for some source X, the algorithms $(\mathsf{E}_X, \mathsf{D}_X)$ implied by $\mathsf{cPREH}^{\mathsf{det}}_{\equiv_s}$ compress that source to its ϵ-smooth min-entropy. To prove this, we introduce the following technical lemma.

LEMMA 17.4 (Splitting lemma). *Let* X *and* Y_x *be distributions giving rise to a joint distribution* (Y, X). *Let* $A \subset \mathit{supp}(Y, X)$ *such that* $\Pr_{(y,x)\leftarrow(Y,X)}[(y,x) \in A] \geqslant 1 - \mu(\lambda)$ *for some negligible function* μ. *Further, for all* $x \in \mathit{supp}(X)$, *let* $p_x := \Pr_{y\leftarrow Y_x}[(y,x) \in A]$. *Then, there exist negligible functions* ν, ν' *and a set* $G \subseteq \mathit{supp}(X)$, *such that for all* $x \in G$, $p_x \geqslant 1 - \nu(\lambda)$ *and* $\Pr_{x\leftarrow X}[x \in G] \geqslant 1 - \nu'(\lambda)$.

Proof. Let $\mu(\lambda)$ be some negligible function. Define $G_n := \{x \in \mathit{supp}(X) \colon p_x \geqslant 1 - n \cdot \mu(\lambda)\}$. Then,

$$
\begin{aligned}
1 - \mu(\lambda) &\leqslant \Pr_{(y,x)\leftarrow(Y,X)}[(y,x) \in A] \\
&= \sum_{x\in G_n} \Pr[X = x] \cdot \underbrace{p_x}_{\leqslant 1} + \sum_{x\notin G_n} \Pr[X = x] \cdot \underbrace{p_x}_{< 1 - n\cdot\mu(\lambda)} \\
&< \Pr_{x\leftarrow X}[x \in G_n] + (1 - n \cdot \mu(\lambda)) \cdot \underbrace{\sum_{x\notin G_n} \Pr[X = x]}_{= 1 - \Pr_{x\leftarrow X}[x\in G_n]} \\
&= \Pr_{x\leftarrow X}[x \in G_n] + 1 - n \cdot \mu(\lambda) - \Pr_{x\leftarrow X}[x \in G_n] \\
&\quad + n \cdot \mu(\lambda) \cdot \Pr_{x\leftarrow X}[x \in G_n] \\
&= 1 - \mu(\lambda) \cdot \left(n - n \cdot \Pr_{x\leftarrow X}[x \in G_n]\right).
\end{aligned}
$$

Hence,

$$\Pr_{x\leftarrow X}\left[x \in G_n\right] \geqslant 1 - \frac{1}{n}.$$

Without loss of generality, we assume $\mu(\lambda) > 0$ for all $\lambda \in \mathbb{N}$. (If $\mu(\lambda) = 0$ for some λ, then for $G := \mathit{supp}(X)$, $p_x = 1$ for all $x \in G$, and $\Pr_{x\leftarrow X}[x \in G] = 1$.) Then, $\sqrt{\mu(\lambda)}$ is well-defined and negligible. Let $n := \sqrt{\mu(\lambda)}^{-1}$. Then, by definition of G_n, for all $x \in G_n$, $p_x \geqslant 1 - n \cdot \mu(\lambda) = 1 - \sqrt{\mu(\lambda)}$. Furthermore, $\Pr_{x\leftarrow X}[x \in G_n] \geqslant 1 - \frac{1}{n} = 1 - \sqrt{\mu(\lambda)}$. □

THEOREM 17.2. *If (weak)* $\mathsf{cPREH}^{\mathsf{det}}_{\equiv_s}$ *is true for* X, *i. e., there exist a setup algorithm* Setup_X *and deterministic algorithms* $(\mathsf{E}_X, \mathsf{D}_X)$ *with* E_X *having output length* n *satisfying correctness and pseudorandomness. Then, the* ϵ*-smooth min-entropy* $H^\epsilon_\infty(X) \geqslant n$ *for some negligible function* ϵ.

Proof. Note that $\mathrm{H}^{\epsilon}_{\infty}(X) = \widetilde{\mathrm{H}}^{\epsilon}_{\infty}(X \mid \mathsf{Setup}_X(1^\lambda))$ as the distribution X is independent of the CRS. Hence, it suffices to upper bound the average ϵ-smooth min-entropy $\widetilde{\mathrm{H}}^{\epsilon}_{\infty}(X \mid \mathsf{Setup}_X(1^\lambda))$.

The proof is similar to the proof of Theorem 17.1. We consider the distribution $Y'_{crs} \coloneqq \mathsf{D}_X(crs, U_n)$. Due to Theorem 16.1, the distributions

$$\{crs \leftarrow \mathsf{Setup}_X(1^\lambda), u \leftarrow U_n \colon (crs, \mathsf{D}_X(crs, u))\} \text{ and}$$
$$\{crs \leftarrow \mathsf{Setup}_X(1^\lambda), y \leftarrow X \colon (crs, y)\}$$

are statistically indistinguishable. Hence,

$$\Delta\big((X \mid \mathsf{Setup}_X(1^\lambda)), (Y' \mid \mathsf{Setup}_X(1^\lambda))\big) \leqslant \delta(\lambda)$$

for some negligible function δ.

Further, by a similar argument as in Theorem 17.1, due to correctness and pseudorandomness we have

$$\Pr\left[crs \leftarrow \mathsf{Setup}_X(1^\lambda), u \leftarrow U_n \colon \mathsf{E}_X\big(crs, \mathsf{D}_X(crs, u)\big) = u\right] \geqslant 1 - \nu'(\lambda)$$

for some negligible function ν', where the probability is over the choice of crs and u.

The Splitting lemma (Lemma 17.4) implies that there exists a set $G \subseteq \mathit{supp}(\mathsf{Setup}_X(1^\lambda))$ such that $\Pr[crs \leftarrow \mathsf{Setup}_X(1^\lambda) \colon crs \in G] \geqslant 1 - \nu''(\lambda)$ and for all $crs \in G$, $\Pr[u \leftarrow U_n \colon \mathsf{E}_X(crs, \mathsf{D}_X(crs, u)) = u] \geqslant 1 - \nu'''(\lambda)$ for some negligible functions ν'', ν'''. Hence, conditioned on $crs \in G$, $\mathsf{D}_X(crs, \cdot)$ operates almost injectively on the set $\{0,1\}^n$. More formally, let $V_0^{crs} \subset \{0,1\}^n$ denote the set of all u such that $\mathsf{E}_X(crs, \mathsf{D}_X(crs, u)) = u$ and let $V_1^{crs} \coloneqq \{0,1\}^n \setminus V_0^{crs}$. For $crs \in G$, we have $|V_1^{crs}|/|\{0,1\}^n| \leqslant \nu'''(\lambda)$. Let $\overline{V_1^{crs}}$ be some arbitrary subset of $\{0,1\}^{k(\lambda)} \setminus \mathsf{D}_X(crs, V_0^{crs})$ such that $|\overline{V_1^{crs}}| = |V_1^{crs}|$ (for some polynomial k). Let Y'' be the uniform distribution over $\mathsf{D}_X(V_0^{crs}) \cup \overline{V_1^{crs}}$. Note that $|\mathsf{D}_X(crs, V_0^{crs}) \cup \overline{V_1^{crs}}| = 2^n$, and for $crs \in G$ we have $\Pr[Y'_{crs} \in \overline{V_1^{crs}}] \leqslant \Pr[Y'_{crs} \notin \mathsf{D}_X(crs, V_0^{crs})] \leqslant \nu'(\lambda)$. We have that

$$\begin{aligned}
&\Delta\big((Y', \mathsf{Setup}_X), (Y'', \mathsf{Setup}_X)\big) \\
&= \sum_{\substack{crs \in \mathit{supp}(\mathsf{Setup}_X) \\ a \in \{0,1\}^{k(\lambda)}}} \Big| \Pr\left[(Y', \mathsf{Setup}_X) = (a, crs)\right] - \Pr\left[(Y'', \mathsf{Setup}_X) = (a, crs)\right]\Big| \\
&= \sum_{crs \in G} \Pr[\mathsf{Setup}_X = crs] \cdot \sum_{a \in \{0,1\}^{k(\lambda)}} \Big|\Pr[Y'_{crs} = a] - \Pr[Y''_{crs} = a]\Big| \\
&\quad + \sum_{crs \notin G} \Pr[\mathsf{Setup}_X = crs] \cdot \sum_{a \in \{0,1\}^{k(\lambda)}} \Big|\Pr[Y'_{crs} = a] - \Pr[Y''_{crs} = a]\Big|
\end{aligned} \tag{17.1}$$

The separate terms can be upper bounded as follows.

$$\begin{aligned}
&\sum_{crs \in G} \Pr\left[\mathsf{Setup}_X = crs\right] \cdot \sum_{a \in \{0,1\}^{k(\lambda)}} \Big|\Pr\left[Y'_{crs} = a\right] - \Pr\left[Y''_{crs} = a\right]\Big| \\
&= \sum_{crs \in G} \Pr\left[\mathsf{Setup}_X = crs\right] \cdot \\
&\qquad \left(\sum_{a \in \mathsf{D}_X(crs, V_0^{crs})} \Big|\Pr\left[Y'_{crs} = a\right] - \Pr\left[Y''_{crs} = a\right]\Big| + \sum_{a \in \overline{V_1^{crs}}} \Big|\Pr\left[Y'_{crs} = a\right] - \Pr\left[Y''_{crs} = a\right]\Big| \right)
\end{aligned}$$

$$\leqslant \sum_{crs \in G} \Pr\left[\mathsf{Setup}_X = crs\right] \cdot \left(1 - \frac{|V_0^{crs}|}{2^n} + \Pr\left[Y'_{crs} \in \overline{V_1^{crs}}\right] + \Pr\left[Y''_{crs} \in \overline{V_1^{crs}}\right]\right)$$

$$\leqslant \sum_{crs \in G} \Pr\left[\mathsf{Setup}_X = crs\right] \cdot \left(\frac{|V_1^{crs}|}{2^n} + \Pr\left[Y'_{crs} \notin \mathsf{D}_X(crs, V_0^{crs})\right] + \frac{|\overline{V_1^{crs}}|}{2^n}\right)$$

$$\leqslant 3 \cdot \nu'''(\lambda) \tag{17.2}$$

$$\sum_{crs \notin G} \Pr\left[\mathsf{Setup}_X = crs\right] \cdot \sum_{a \in \{0,1\}^{k(n)}} \left|\Pr\left[Y'_{crs} = a\right] - \Pr\left[Y''_{crs} = a\right]\right|$$

$$\leqslant \sum_{crs \notin G} \Pr\left[\mathsf{Setup}_X = crs\right] \cdot 2$$

$$\leqslant 2 \cdot \nu''(\lambda) \tag{17.3}$$

Combining Equations (17.1), (17.2) and (17.3) we have

$$\Delta\big((Y', \mathsf{Setup}_X), (Y'', \mathsf{Setup}_X)\big) \leqslant 2 \cdot \nu''(\lambda) + 3 \cdot \nu'''(\lambda).$$

Furthermore, since for all $crs \in supp(\mathsf{Setup}_X(1^\lambda))$, Y''_{crs} is the uniform distribution over a set of 2^n elements,

$$\widetilde{H}_\infty\big(Y'' \mid \mathsf{Setup}_X(1^\lambda)\big) = -\log\left(\mathop{\mathbb{E}}_{crs \leftarrow \mathsf{Setup}_X(1^\lambda)} \max_a \Pr[Y''_{crs} = a]\right)$$

$$= n.$$

Therefore, we have

$$H_\infty^{\delta+2\nu''+3\nu'''}(X) = \widetilde{H}_\infty^{\delta+2\nu''+3\nu'''}(X \mid \mathsf{Setup}_X(1^\lambda)) \geqslant n,$$

where δ, ν'', ν''' are negligible functions. □

Theorem 17.2 together with Lemmas 15.1 and 17.3 yield the following corollary.

COROLLARY 17.3. *If one-way functions exist,* $\mathsf{cPREH}^{\mathsf{det}}_{\equiv_s}$ *is false.*

17.1.2 Computational Guarantees and Pseudoentropy

We study the relation of the pseudorandom encoding hypothesis with deterministic encoding algorithm to HILL and Yao entropy. Interestingly, $\mathsf{PREH}^{\mathsf{det}}_{\approx_c}$ poses a lower bound on the HILL entropy of a source and an upper bound on its Yao entropy.

HILL ENTROPY. The algorithms $(\mathsf{E}_X, \mathsf{D}_X)$ implied by $\mathsf{PREH}^{\mathsf{det}}_{\approx_c}$ compress a source X to its HILL entropy.

THEOREM 17.3. *If (weak)* $\mathsf{PREH}^{\mathsf{det}}_{\approx_c}$ *is true for* X*, i. e., there exist deterministic algorithms* $(\mathsf{E}_X, \mathsf{D}_X)$ *with* E_X *having output length* n *satisfying correctness and pseudorandomness. Then,* $H^{\mathsf{HILL}}(X) \geqslant n$.

Proof. We employ a similar strategy as in the proof of Theorem 17.1. Let $Y' \coloneqq \mathsf{D}_X(U_n)$. As already seen in the proof of Theorem 16.1, Y' and X are

computationally indistinguishable due to correctness and pseudorandomness.

Due to correctness, $\Pr[x \leftarrow X\colon \mathsf{D}_X(\mathsf{E}_X(x)) = x] \geqslant 1 - \nu(\lambda)$ for some negligible function ν. Since E_X is required to be deterministic, we get $\Pr[x \leftarrow X\colon \mathsf{E}_X(\mathsf{D}_X(\mathsf{E}_X(x))) = \mathsf{E}_X(x)] \geqslant 1 - \nu(\lambda)$. Applying pseudorandomness, we have

$$\Pr\left[u \leftarrow U_n\colon \mathsf{E}_X\big(\mathsf{D}_X(u)\big) = u\right] \geqslant 1 - \nu'(\lambda)$$

for some negligible function ν', where the probability is only over the choice of u. Let $V_0 \subset \{0,1\}^n$ denote the set of all u such that $\mathsf{E}_X(\mathsf{D}_X(u)) = u$ and let $V_1 \coloneqq \{0,1\}^n \setminus V_0$. We have that, $|V_1|/|\{0,1\}^n| \leqslant \nu'(\lambda)$.

Let $\overline{V_1}$ be some arbitrary subset of $\{0,1\}^{k(\lambda)} \setminus \mathsf{D}_X(V_0)$ such that $|\overline{V_1}| = |V_1|$ (for some polynomial k). Let Y'' be the uniform distribution over $\mathsf{D}_X(V_0) \cup \overline{V_1}$. The min-entropy of Y'' equals n. By the same argument as in the proof of Theorem 17.1, the statistical distance between Y' and Y'' is negligible.

Therefore, the distributions X and Y'' are computationally indistinguishable and, hence, $\mathrm{H}^{\mathsf{HILL}}(X) \geqslant n$. □

This result can be generalized to conditional HILL entropy using general (as opposed to weak) $\mathsf{PREH}^{\mathsf{det}}_{\approx_c}$.

THEOREM 17.4. *Let (X, Z) be a joint distribution. More precisely, let Z be a distribution over words of length λ. For $z \in \mathit{supp}(Z)$, let X_z denote the conditional distribution when $Z = z$. If (general) $\mathsf{PREH}^{\mathsf{det}}_{\approx_c}$ is true for X, i.e., there exist two deterministic polynomial time algorithms $(\mathsf{E}_X, \mathsf{D}_X)$ with E_X having output length n satisfying correctness and pseudorandomness. Then, $\mathrm{H}^{\mathsf{HILL}}(X \mid Z) \geqslant n$.*

Proof. For each $z \in \mathit{supp}(Z)$, consider the distribution $Y'_z \coloneqq \mathsf{D}_X(z, U_n)$. Due to Theorem 16.1, for all adversarially chosen z, the distributions $\{(Y'_z, z)\}$ and $\{(X_z, z)\}$ are computationally indistinguishable due to correctness and pseudorandomness.

Due to correctness, for all adversarially chosen z, the probability $\Pr[x \leftarrow X_z\colon \mathsf{D}_X(z, \mathsf{E}_X(z, x)) = x] \geqslant 1 - \nu(\lambda)$ for some negligible function ν. Since E_X is required to be deterministic, we have that for all adversarially chosen z, $\Pr[x \leftarrow X_z\colon \mathsf{E}_X(z, \mathsf{D}_X(z, \mathsf{E}_X(z, x))) = \mathsf{E}_X(z, x)] \geqslant 1 - \nu(\lambda)$. Applying pseudorandomness, we get that there exists a negligible function ν' such that for all adversarially chosen z,

$$\Pr\left[u \leftarrow U_n\colon \mathsf{E}_X\big(z, \mathsf{D}_X(z, u)\big) = u\right] \geqslant 1 - \nu'(\lambda),$$

where the probability is only over the choice of u. Let $V_{0,z} \subseteq \{0,1\}^n$ denote the set of all $u \in \{0,1\}^n$ such that $\mathsf{E}_X(z, \mathsf{D}_X(z, u)) = u$ holds and let $V_{1,z} \coloneqq \{0,1\}^n \setminus V_{0,z}$. We have that $|V_{1,z}|/|\{0,1\}^n| \leqslant \nu'(\lambda)$.

Let $\overline{V_{1,z}}$ be some arbitrary subset of $\{0,1\}^{k(\lambda)} \setminus \mathsf{D}_X(z, V_{0,z})$ such that $|\overline{V_{1,z}}| = |V_{1,z}|$ (for some polynomial k). Let Y''_z be the uniform distribution over $\mathsf{D}_X(z, V_{0,z}) \cup \overline{V_{1,z}}$. Clearly,

$$\begin{aligned}\widetilde{\mathrm{H}}_\infty(Y'' \mid Z) &= -\log\Big(\mathop{\mathbb{E}}_{z \leftarrow Z}\big[\max_{y \in \mathit{supp}(Y''_z)} \Pr[Y''_z = y]\big]\Big)\\ &= n.\end{aligned}$$

By a similar argument as in the proof of Theorem 17.1, for all adversarially chosen z, the statistical distance between Y'_z and Y''_z is negligible:

$$\begin{aligned}\Delta(Y'_z, Y''_z) = &\underbrace{\sum_{a \in D_X(z, V_{z,0})} \left|\Pr[Y'_z = a] - \Pr[Y''_z = a]\right|}_{\leqslant 1 - |V_{0,z}| \cdot 2^{-n} \leqslant 1 - (1 - \nu'(\lambda))} \\ &+ \underbrace{\sum_{a \in \overline{V_{1,z}}} \left|\Pr[Y' = a] - \Pr[Y'' = a]\right|}_{\leqslant \sum_{a \in \overline{V_{1,z}}} (\Pr[Y' = a] + \Pr[Y'' = a]) \leqslant 2 \cdot \nu'(\lambda)} \\ &\leqslant 3 \cdot \nu'(\lambda)\end{aligned}$$

Hence, the (joint) distributions (X, Z) and (Y'', Z) are computationally indistinguishable. Therefore, $H^{\mathsf{HILL}}(X \mid Z) \geqslant n$. □

We recall that in the proof of Theorem 17.2, we used the observation that independent random variables A and B satisfy $H_\infty(A) = H_\infty(A \mid B)$. However, this does not necessarily extend to the computational case, meaning that in our case $H^{\mathsf{HILL}}(X \mid \mathsf{Setup}_X(1^\lambda), Z)$ and $H^{\mathsf{HILL}}(X \mid Z)$ are not equal even though the random variables X and Setup_X are independent. Hence, in order to extend Theorem 17.4 to $\mathsf{cPREH}^{\mathsf{det}}_{\approx_c}$, we need that there exists a "good" CRS crs such that $Y'_{crs,z}$ is almost injective on $\{0,1\}^n$ and such that $\{(crs, y)\}$ and $\{(crs, D_X(crs, z, u))\}$ are computationally indistinguishable for this fixed crs. However, since the adversary is quantified after the crs, a non-uniform adversary could know the randomness which was used to generate crs compromising all security guarantees. Therefore, we can only hope to obtain an upper bound on n depending on the *conditional* HILL entropy $H^{\mathsf{HILL}}(X \mid \mathsf{Setup}_X(1^\lambda), Z)$.

THEOREM 17.5. *Let (X, Z) be a joint distribution. More precisely, let Z be a distribution over words of length λ. For $z \in supp(Z)$, let X_z denote the conditional distribution when $Z = z$. If (general) $\mathsf{cPREH}^{\mathsf{det}}_{\approx_c}$ is true for X, i. e., there exists a PPT algorithm Setup_X and two deterministic polynomial time algorithms (E_X, D_X) with E_X having output length n satisfying correctness and pseudorandomness. Then, $H^{\mathsf{HILL}}(X \mid \mathsf{Setup}_X(1^\lambda), Z) \geqslant n$.*

Proof. The proof is similar to the proof of Theorem 17.4. We consider the distribution $Y'_{crs,z} \coloneqq D_X(crs, z, U_n)$.

Due to Theorem 16.1, we have that for all adversarially chosen z, the distributions $\{crs \leftarrow \mathsf{Setup}_X(1^\lambda), y \leftarrow X_z : (crs, z, y)\}$ and $\{crs \leftarrow \mathsf{Setup}_X(1^\lambda), u \leftarrow U_n : (crs, z, D_X(crs, z, u))\}$ are computationally indistinguishable.

Due to correctness and pseudorandomness we have that there exists a negligible function ν', such that for all adversarially chosen z,

$$\begin{aligned}&\Pr\left[crs \leftarrow \mathsf{Setup}_X(1^\lambda), u \leftarrow U_n : E_X\big(crs, z, D_X(crs, z, u)\big) = u\right] \\ &\geqslant 1 - \nu'(\lambda),\end{aligned}$$

where the probability is over the choice of crs and u. The Splitting lemma (Lemma 17.4) implies that for all z,[62] there exists a set $G_z \subseteq supp(\mathsf{Setup}_X(1^\lambda))$ such that $\Pr[crs \leftarrow \mathsf{Setup}_X(1^\lambda) : crs \in G_z] \geqslant 1 - \nu''_z(\lambda)$ and for all $crs \in G_z$, $\Pr[u \leftarrow U_n : E_X(crs, z, D_X(crs, z, u)) = u] \geqslant 1 - \nu'''_z(\lambda)$ for some negligible functions ν''_z, ν'''_z. Hence, conditioned on $crs \in G_z$, $D_X(crs, z, \cdot)$ operates almost injectively on the set $\{0,1\}^n$.

[62] *Note that here we use that the adversary is non-uniform.*

Let $V^{crs}_{0,z} \subseteq \{0,1\}^n$ denote the set of all $u \in \{0,1\}^n$ such that $\mathsf{E}_X(crs, z, \mathsf{D}_X(crs, z, u)) = u$ holds and let $V^{crs}_{1,z} := \{0,1\}^n \setminus V^{crs}_{0,z}$. For $crs \in \mathsf{G}_z$, we have $|V^{crs}_{1,z}|/|\{0,1\}^n| \leqslant \nu'''_z(\lambda)$.

Let $\overline{V^{crs}_{1,z}}$ be some arbitrary subset of $\{0,1\}^{k(\lambda)} \setminus \mathsf{D}_X(crs, z, V^{crs}_{0,z})$ such that $|\overline{V^{crs}_{1,z}}| = |V^{crs}_{1,z}|$ (for some polynomial k). Let $Y''_{crs,z}$ be the uniform distribution over $\mathsf{D}_X(crs, z, V^{crs}_{0,z}) \cup \overline{V^{crs}_{1,z}}$. Then,

$$\begin{aligned}
&\widetilde{H}_\infty\big(Y'' \;\big|\; \mathsf{Setup}_X(1^\lambda), Z\big) \\
&\quad = -\log\left(\mathop{\mathbb{E}}_{\substack{crs \leftarrow \mathsf{Setup}_X(1^\lambda) \\ z \leftarrow Z}}\left[\max_{y \in supp(Y''_{crs,z})} \Pr\left[Y''_{crs,z} = y\right]\right]\right) \\
&\quad = n.
\end{aligned}$$

Furthermore, a similar computation as in the proof of Theorem 17.2, for all adversarially chosen z, the statistical distance between $(Y'_z, \mathsf{Setup}_X(1^\lambda))$ and $(Y''_z, \mathsf{Setup}_X(1^\lambda))$ is negligible. Hence, the (joint) distributions $(X, \mathsf{Setup}_X(1^\lambda), Z)$ and $(Y'', \mathsf{Setup}_X(1^\lambda), Z)$ are computationally indistinguishable. Therefore, $H^{\mathsf{HILL}}(X \mid \mathsf{Setup}_X(1^\lambda), Z) \geqslant n$.

□

YAO ENTROPY. On the other hand, the existence of algorithms $(\mathsf{E}_X, \mathsf{D}_X)$ as implied by $\mathsf{PREH}^{\mathsf{det}}_{\approx_c}$ give an upper bound for the Yao entropy of a source.

LEMMA 17.5. *If (weak)* $\mathsf{PREH}^{\mathsf{det}}_{\approx_c}$ *is true for* X*, i.e., there exist deterministic algorithms* $(\mathsf{E}_X, \mathsf{D}_X)$ *with* E_X *having output length* n *satisfying correctness and pseudorandomness. Then,* $H^{\mathsf{Yao}}(X) < n + \delta(\lambda)$ *for some negligible* δ*.*

Proof. Due to correctness, there exists a pair of efficient algorithms $(\mathsf{E}_X, \mathsf{D}_X)$ (with E_X having output length n) such that

$$\Pr_{x \leftarrow X}\left[\mathsf{D}_X(\mathsf{E}_X(x)) = x\right] \geqslant 1 - \nu(\lambda)$$

for some negligible function ν. Hence, by Definition 17.4, $H^{\mathsf{Yao}}(X) < k$, for all k satisfying

$$\begin{aligned}
& 1 - \nu(\lambda) \geqslant 2^{n-k} + \epsilon(\lambda) \\
\iff\quad & 2^k \cdot (1 - \nu(\lambda) - \epsilon(\lambda)) \geqslant 2^n \\
\iff\quad & k \geqslant n - \log\big(1 - \nu(\lambda) - \epsilon(\lambda)\big).
\end{aligned}$$

□

This result can be generalized to conditional Yao entropy as follows.

LEMMA 17.6. *Let* (X, Z) *be a joint distribution. More precisely, let* Z *be a distribution over words of length* λ*. For* $z \in supp(Z)$*, let* X_z *denote the conditional distribution when* $Z = z$*. If (general)* $\mathsf{PREH}^{\mathsf{det}}_{\approx_c}$ *is true for* X*, i.e., there exist two deterministic polynomial time algorithms* $(\mathsf{E}_X, \mathsf{D}_X)$ *with* E_X *having output length* n *satisfying correctness and pseudorandomness. Then,* $H^{\mathsf{Yao}}(X \mid Z) < n + \delta(\lambda)$ *for some negligible* δ*.*

Proof. Due to correctness, for all adversarially chosen z,

$$\Pr_{x \leftarrow X_z}\left[\mathsf{D}_X(z, \mathsf{E}_X(z, x)) = x\right] \geqslant 1 - \nu(\lambda)$$

for some negligible function ν. Hence, by Definition 17.6, $H^{Yao}(X \mid Z) < k$, for all k satisfying

$$k \geqslant n - \log\left(1 - \nu(\lambda) - \epsilon(\lambda)\right).$$

□

This result can further be extended to the case of a common setup.

LEMMA 17.7. *Let (X, Z) be a joint distribution. More precisely, let Z be a distribution over words of length λ. For $z \in supp(Z)$, let X_z denote the conditional distribution when $Z = z$. If (general) $\mathsf{cPREH}^{det}_{\approx_c}$ is true for X, i.e., there exists a setup algorithm Setup and two deterministic polynomial time algorithms $(\mathsf{E}_X, \mathsf{D}_X)$ with E_X having output length n satisfying correctness and pseudorandomness. Then, $H^{Yao}(X \mid Z) < n + \delta(\lambda)$ for some negligible δ.*

Proof. The Splitting lemma (Lemma 17.4) implies that for all z,[63] there is a set $G_z \subseteq supp(\mathsf{Setup}_X(1^\lambda))$ such that $\Pr[crs \leftarrow \mathsf{Setup}_X(1^\lambda) : crs \in G_z] \geqslant 1 - \nu''_z(\lambda)$ and for all $crs \in G_z$,

$$\Pr\left[u \leftarrow U_n : \mathsf{E}_X\left(crs, z, \mathsf{D}_X(crs, z, u)\right) = u\right] \geqslant 1 - \nu'''_z(\lambda)$$

for some negligible functions ν''_z, ν'''_z.

We exploit the non-uniformity of the definition (Definitions 17.4 and 17.6). In particular, we define the compression and decompression circuits as $\mathsf{E}'_X(y, z; aux_\lambda) := \mathsf{E}_X(aux_\lambda, z, y)$ and $\mathsf{D}'_X(u, z; aux_\lambda) := \mathsf{D}_X(aux_\lambda, z, u)$, where aux_λ denotes the non-uniform auxiliary input providing a "good" CRS. □

[63] *Note that here we use that the adversary is non-uniform.*

Due to [HLR07], assuming suitable non-interactive zero-knowledge proof systems and pseudorandom generators, there exists a joint distribution (X, Z) with high conditional Yao entropy but low conditional HILL entropy. The sampler S from [HLR07] takes as input a NIZK common random string σ, samples a seed from U_λ and outputs $y_1 := \mathsf{PRG}(s)$ together with a proof y_2 that y_1 is in the image of PRG. Due to Lemma 15.2, $S \in \mathcal{S}^{\approx_c}$. This yields the following corollary.

COROLLARY 17.4. *If there exist a pseudorandom generator and a (single-theorem) NIZK proof system such that (i) for an overwhelming fraction of common random strings, the number of accepting proofs for each statement is limited, and (ii) the simulated random string is independent of the statement as in [HLR07]. Then $\mathsf{PREH}^{det}_{\approx_c}$ is false.*

A Blum integer is a natural number $N = p \cdot q$ such that p, q are primes with $p \equiv q \equiv 3 \bmod 4$. The quadratic residuosity assumption (QRA) states that for a randomly chosen Blum integer $N = p \cdot q$, the distributions $\{y \leftarrow \mathbb{Z}^\times_N \text{ s.t. } \left(\frac{y}{p}\right) = \left(\frac{y}{q}\right) = 1 : (N, y)\}$ and $\{y \leftarrow \mathbb{Z}^\times_N \text{ s.t. } \left(\frac{y}{N}\right) = 1 : (N, y)\}$ are computationally indistinguishable.

Due to [LMS05], a (single-theorem) NIZK proof system which is suitable for Corollary 17.4 can by instantiated from the quadratic residuosity assumption (QRA).

COROLLARY 17.5. *If the quadratic residuosity assumption is true, then $\mathsf{PREH}^{det}_{\approx_c}$ is false.*

ON REFUTING $\mathsf{cPREH}^{\mathsf{det}}_{\approx_c}$. Theorem 17.5 only yields an upper bound on the encoding length n depending on $H^{\mathsf{HILL}}(X \mid \mathsf{Setup}_X(1^\lambda), Z)$. Since the Yao entropy of any source $H^{\mathsf{Yao}}(X \mid \mathsf{Setup}_X(1^\lambda), Z)$ is upper bounded by $n + \mathit{negl}(\lambda)$, we cannot apply the result from [HLR07] to refute $\mathsf{cPREH}^{\mathsf{det}}_{\approx_c}$.

ON REFUTING WEAK $\mathsf{PREH}^{\mathsf{det}}_{\approx_c}$. It is currently not known if plain HILL and plain Yao entropy can be separated as in [HLR07]. Such a separation would refute weak $\mathsf{PREH}^{\mathsf{det}}_{\approx_c}$.

17.2 RANDOMIZED ENCODING ALGORITHM

In the following we prove positive and negative results on the validity of the pseudorandom encoding hypothesis with a randomized encoding algorithm. In particular, in Section 17.2.2 we refute $\mathsf{PREH}^{\mathsf{rand}}_{\equiv_s}$ based on subexponential LWE, in Section 17.2.3 we refute $\mathsf{cPREH}^{\mathsf{rand}}_{\approx_c}$ based on extractable one-way functions with *unbounded* auxiliary input. On the positive side, in Section 17.3, we give a construction of perfectly correct $\mathsf{cPREH}^{\mathsf{rand}}_{\approx_c}$ with and without universal setup based on indistinguishability obfuscation and one-way functions following [SW14; DKR15] together with Theorem 16.1. In Section 17.4, we bootstrap $\mathsf{cPREH}^{\mathsf{rand}}_{\approx_c}$ with a common random string from the construction in Chapter 6 additionally assuming *weak* $\mathsf{cPREH}^{\mathsf{rand}}_{\approx_c}$ with a common random string. Since $\mathsf{cPREH}^{\mathsf{rand}}_{\approx_c}$ with a common random string in conjunction with NIZK proof systems contradicts EOWFs with common but benign auxiliary information, this refutes even *weak* $\mathsf{cPREH}^{\mathsf{rand}}_{\approx_c}$ with common random string.

17.2.1 (Generalized) Extractable One-way Functions

In this section, we define generalized extractable one-way functions (with respect to common auxiliary input) as in [BCPR16].

DEFINITION 17.9 (Generalized extractable one-way function family ensembles with common auxiliary information, [BCPR16]). A *generalized one-way extractable function (GEOWF) family ensemble with common auxiliary information*, with respect to a relation $\mathcal{R}^{\mathcal{F}}_f$ over triplets $(f, y, x) \in \mathsf{Gen}(1^\lambda) \times \{0,1\}^{n(\lambda)} \times \{0,1\}^{m(\lambda)}$, is a function family ensemble $\mathcal{F} = (\mathsf{Gen}, \mathsf{Eval})$ (cf. Definition 2.8) if the following properties are satisfied.

$\mathcal{R}^{\mathcal{F}}$**-HARDNESS.** For every PPT adversary $\mathcal{A}$, for every polynomial b and every $z \in \{0,1\}^{b(\lambda)}$,

$$\Pr\left[\mathrm{Exp}^{\text{g-hard-aux}}_{\mathcal{F},\mathcal{A},z}(\lambda) = 1\right]$$

is negligible, where $\mathrm{Exp}^{\text{g-hard-aux}}_{\mathcal{F},\mathcal{A},z}(\lambda)$ is defined in Figure 17.1a.

$\mathcal{R}^{\mathcal{F}}$**-EXTRACTABILITY.** For every PPT adversary $\mathcal{X}$, there exists a PPT algorithm $\mathcal{E}_{\mathcal{X}}$ such that for every polynomial b and every $z \in \{0,1\}^{b(\lambda)}$,

$$\Pr\left[\mathrm{Exp}^{\text{g-ext-aux}}_{\mathcal{F},\mathcal{X},\mathcal{E}_{\mathcal{X}},z}(\lambda) = 1\right]$$

is overwhelming, where $\mathrm{Exp}^{\text{g-ext-aux}}_{\mathcal{F},\mathcal{X},\mathcal{E}_{\mathcal{X}},z}(\lambda)$ is defined in Figure 17.1b.

We call $\mathcal{F}$

- *publicly verifiable* if there exists a PPT algorithm T such that

$$\mathsf{T}(f, f(x), x') = 1 \Leftrightarrow \mathcal{R}_f^{\mathcal{F}}(f(x), x') = 1.$$

- *privately verifiable* if there exists a PPT algorithm T such that

$$\mathsf{T}(f, x, x') = 1 \Leftrightarrow \mathcal{R}_f^{\mathcal{F}}(f(x), x') = 1.$$

$\mathrm{Exp}^{\text{g-hard-aux}}_{\mathcal{F},\mathcal{A},z}(\lambda)$

$f \leftarrow \mathsf{Gen}(1^\lambda)$

$x \leftarrow \{0,1\}^{n(\lambda)}$

$y := f(x)$

$x' \leftarrow \mathcal{A}(f, y, z)$

return $\mathcal{R}_f^{\mathcal{F}}(y, x')$

(a) The $\mathcal{R}^{\mathcal{F}}$-hardness game.

$\mathrm{Exp}^{\text{g-ext-aux}}_{\mathcal{F},\mathcal{X},\mathcal{E}_{\mathcal{X}},z}(\lambda)$

$f \leftarrow \mathsf{Gen}(1^\lambda)$

$r_{\mathcal{X}} \leftarrow \{0,1\}^{p(\lambda)}$

$y \leftarrow \mathcal{X}(f, z; r_{\mathcal{X}})$

$x \leftarrow \mathcal{E}_{\mathcal{X}}(f, z, r_{\mathcal{X}})$

return $\left(\mathcal{R}_f^{\mathcal{F}}(y, x)\right) \vee \left(\forall x' \colon f(x') \neq y\right)$

(b) The $\mathcal{R}^{\mathcal{F}}$-extraction game.

Figure 17.1: $\mathcal{R}^{\mathcal{F}}$-hardness and $\mathcal{R}^{\mathcal{F}}$-extraction game for GEOWFs (with common auxiliary input).

DEFINITION 17.10 (Generalized extractable one-way function family ensembles with b-bounded common auxiliary information, [BCPR16]). Like Definition 17.9 but with a fixed polynomial b determining the length of the common auxiliary information.

Due to [BCPR16], privately verifiable generalized extractable one-way functions with bounded auxiliary input can be instantiated based on falsifiable assumptions.

THEOREM 17.6 ([BCPR16]). *Assuming the learning with errors problem is subexponentially hard, then there exists a* $(b(\lambda) - \omega(1))$*-bounded privately verifiable GEOWF family ensemble.*

17.2.2 Information-theoretic Guarantees and Privately Verifiable GEOWFs

Assuming a pseudorandom encoding scheme providing information-theoretic guarantees, we can consider unbounded adversaries. In this setting, private verifiability is not a restriction.

THEOREM 17.7. *If privately verifiable generalized extractable one-way functions without auxiliary information exist, then* $\mathsf{PREH}^{\mathsf{rand}}_{\equiv_s}$ *is false.*

Proof. The proof strategy follows the ideas of [IKOS10]. Let $\mathcal{F}$ be a privately verifiable GEOWF (without auxiliary input) with respect to relation $\mathcal{R}^{\mathcal{F}}$.

$\mathsf{PREH}^{\mathsf{rand}}_{\equiv_s}$ implies that for the algorithm S (given in Figure 17.2a) there exist an alternative sampler $\overline{S}$ and a corresponding inverse sampler $\overline{S}^{-1}$ satisfying closeness and invertibility of Definition 16.1 against unbounded adversaries. Since $\mathcal{F}$ is a GEOWF, for the algorithm $\overline{S}$, there exists an extractor $\mathcal{E}_{\overline{S}}$ satisfying extractability from Definition 17.9 (without auxiliary information).

$S(f)$

$x \leftarrow \{0,1\}^{n(\lambda)}$

$y \coloneqq f(x)$

return y

(a) Sampler mapping a random pre-image with f.

$\mathcal{A}(1^\lambda, f, y)$

$r'_{\overline{S}} \leftarrow \overline{S}^{-1}(f, y)$

$x' \leftarrow \mathcal{E}_{\overline{S}}(f, r'_X)$

return x'

(b) An adversary breaking one-wayness.

Figure 17.2: Description of the sampler S and of the adversary $\mathcal{A}$ breaking one-wayness of the privately verifiable GEOWF.

We prove that for $\mathcal{A}$ given in Figure 17.2b,

$$\Pr\left[\mathrm{Exp}^{\text{g-hard}}_{\mathcal{F},\mathcal{A}}(\lambda) = 1\right] \text{ is overwhelming.}$$

Let $\widetilde{T}$ be an *unbounded* algorithm that given (f, y, x), computes the relation $\mathcal{R}^{\mathcal{F}}_f(y, x)$.[64] We proceed over a series of hybrids, see Figure 17.3.

[64] *Since $\mathcal{F}$ is privately verifiable, an efficient algorithm computing the relation $\mathcal{R}^{\mathcal{F}}_f$ additionally requires the pre-image of y as an input. For our purpose, it suffices to consider an inefficient testing algorithm $\widetilde{T}$.*

$\mathbf{G}_0$

$f \leftarrow \mathsf{Gen}(1^\lambda)$

$x \leftarrow \{0,1\}^{n(\lambda)}$

$y \coloneqq f(x)$

$x' \leftarrow \mathcal{A}(f, y)$

return $\widetilde{T}(f, y, x')$

(a) Description of $\mathbf{G}_0$.

$\mathbf{G}_1$

$f \leftarrow \mathsf{Gen}(1^\lambda)$

$x \leftarrow \{0,1\}^{n(\lambda)}$

$y \coloneqq f(x)$

$r'_{\overline{S}} \leftarrow \overline{S}^{-1}(f, y)$

$x' \leftarrow \mathcal{E}_{\overline{S}}(f, r'_{\overline{S}})$

return $\widetilde{T}(f, y, x')$

(b) Description of $\mathbf{G}_1$.

$\mathbf{G}_2$

$f \leftarrow \mathsf{Gen}(1^\lambda)$

$y \leftarrow S(f)$

$r'_{\overline{S}} \leftarrow \overline{S}^{-1}(f, y)$

$x' \leftarrow \mathcal{E}_{\overline{S}}(f, r'_{\overline{S}})$

return $\widetilde{T}(f, y, x')$

(c) Description of $\mathbf{G}_2$.

$\mathbf{G}_3$

$f \leftarrow \mathsf{Gen}(1^\lambda)$

$y \leftarrow \overline{S}(f)$

$r'_{\overline{S}} \leftarrow \overline{S}^{-1}(f, y)$

$x' \leftarrow \mathcal{E}_{\overline{S}}(f, r'_{\overline{S}})$

return $\widetilde{T}(f, y, x')$

(d) Description of $\mathbf{G}_3$.

$\mathbf{G}_4$

$f \leftarrow \mathsf{Gen}(1^\lambda)$

$r_{\overline{S}} \leftarrow \{0,1\}^{n(\lambda)}$

$y \leftarrow \overline{S}(f; r_{\overline{S}})$

$r'_{\overline{S}} \leftarrow \overline{S}^{-1}(f, y)$

$x' \leftarrow \mathcal{E}_{\overline{S}}(f, r'_{\overline{S}})$

return $\widetilde{T}(f, y, x')$

(e) Description of $\mathbf{G}_4$.

$\mathbf{G}_5$

$f \leftarrow \mathsf{Gen}(1^\lambda)$

$r_{\overline{S}} \leftarrow \{0,1\}^{n(\lambda)}$

$y \leftarrow \overline{S}(f; r_{\overline{S}})$

$x' \leftarrow \mathcal{E}_{\overline{S}}(f, r_{\overline{S}})$

return $\widetilde{T}(f, y, x')$

(f) Description of $\mathbf{G}_5$.

Figure 17.3: Hybrids used in the proof of Theorem 17.7.

GAME $\mathbf{G}_0$. $\mathbf{G}_0$ corresponds to $\mathrm{Exp}^{\text{g-hard}}_{\mathcal{F},\mathcal{A}}(\lambda)$ using the inefficient algorithm $\widetilde{T}$ to test whether $(f(x), x') \in \mathcal{R}^{\mathcal{F}}_f$.

GAME $\mathbf{G}_1$. $\mathbf{G}_1$ is identical to $\mathbf{G}_0$ except that the implementation of $\mathcal{A}$ is explicit in $\mathbf{G}_1$. Hence, $\Pr[out_0 = 1] = \Pr[out_1 = 1]$.

GAME $\mathbf{G}_2$. $\mathbf{G}_2$ is identical to $\mathbf{G}_1$ except that $y \coloneqq f(x)$ for $x \leftarrow \{0,1\}^{n(\lambda)}$ is replaced with $y \leftarrow S(f)$ which proceeds identically, see Figure 17.2a. Hence, $\Pr[out_1 = 1] = \Pr[out_2 = 1]$.

GAME $\mathbf{G}_3$. $\mathbf{G}_3$ is identical to $\mathbf{G}_2$ except that $\mathbf{G}_2$ samples y using $S(f)$ whereas $\mathbf{G}_3$ samples y using $\overline{S}(f)$.

CLAIM 17.1. There exists an unbounded adversary $\mathcal{B}_3$ such that $|\Pr[out_3 = 1] - \Pr[out_2 = 1]| \leqslant \mathrm{Adv}^{\mathsf{close}}_{(\overline{S},\overline{S}^{-1}),\mathcal{B}_3,\overline{f}}(\lambda)$ (for some $\overline{f} \in \mathit{supp}(\mathsf{Gen}(1^\lambda))$).

Proof. Construct an adversary $\mathcal{B}_3$ breaking statistical closeness. $\mathcal{B}_3$ receives as input a key $\overline{f}$ and some y that has either been sampled via $S(\overline{f})$ or via $\overline{S}(\overline{f})$. $\mathcal{B}_3$ computes $r'_{\overline{S}}$ and x' as in game $\mathbf{G}_2$, calls the (inefficient) testing algorithm $\widetilde{T}$ on input of $(\overline{f}, y, x')$ and outputs the resulting output. Hence, we have

$$\Pr\left[\mathrm{Exp}^{(0)\text{-}\mathsf{close}}_{(\overline{S},\overline{S}^{-1}),\mathcal{B}_3,\overline{f}}(\lambda) = 1\right] = \Pr\left[out_2 = 1 \,\middle|\, f = \overline{f}\right] \text{ and}$$
$$\Pr\left[\mathrm{Exp}^{(1)\text{-}\mathsf{close}}_{(\overline{S},\overline{S}^{-1}),\mathcal{B}_3,f}(\lambda) = 1\right] = \Pr\left[out_3 = 1 \,\middle|\, f = \overline{f}\right],$$

and, since $\mathrm{Adv}^{\mathsf{close}}_{(\overline{S},\overline{S}^{-1}),\mathcal{B}_3,\overline{f}}(\lambda)$ is negligible for all $\overline{f} \in \mathit{supp}(\mathsf{Gen}(1^\lambda))$,

$$\begin{aligned}&\left|\Pr\left[out_3 = 1\right] - \Pr\left[out_2 = 1\right]\right| \\ &= \sum_{\overline{f} \in \mathit{supp}(\mathsf{Gen}(1^\lambda))} \Pr_{f \leftarrow \mathsf{Gen}(1^\lambda)}[f = \overline{f}] \cdot \mathrm{Adv}^{\mathsf{close}}_{(\overline{S},\overline{S}^{-1}),\mathcal{B}_3,\overline{f}}(\lambda),\end{aligned}$$

is negligible. □

GAME $\mathbf{G}_4$. $\mathbf{G}_4$ is identical to $\mathbf{G}_3$ except that the random coins $r_{\overline{S}}$ used for $\overline{S}$ are made explicit. Hence, $\Pr[out_3 = 1] = \Pr[out_4 = 1]$.

GAME $\mathbf{G}_5$. $\mathbf{G}_5$ is identical to $\mathbf{G}_4$ except that $\mathcal{E}_{\overline{S}}$ is called using $r_{\overline{S}}$ instead of the inverse sampled random coins $r'_{\overline{S}}$.

CLAIM 17.2. There exists an unbounded adversary $\mathcal{B}_5$ such that $|\Pr[out_5 = 1] - \Pr[out_4 = 1]| \leqslant \mathrm{Adv}^{\mathsf{inv}}_{(\overline{S},\overline{S}^{-1}),\mathcal{B}_5,\overline{f}}(\lambda)$ (for some $\overline{f} \in \mathit{supp}(\mathsf{Gen}(1^\lambda))$).

Proof. Construct an unbounded adversary $\mathcal{B}_5$ breaking statistical invertibility. On input of $(\overline{f}, r, y)$, $\mathcal{B}_5$ calls $\mathcal{E}_{\overline{S}}$ on input of $(\overline{f}, r)$ and obtains x'. Finally, $\mathcal{B}_5$ calls the inefficient testing algorithm $\widetilde{T}$ on input of $(\overline{f}, y, x')$ and outputs the resulting output. If $\mathcal{B}_5$ plays the experiment $\mathrm{Exp}^{(0)\text{-}\mathsf{inv}}_{(\overline{S},\overline{S}^{-1}),\mathcal{B}_5,\overline{f}}(\lambda)$, he perfectly simulates game $\mathbf{G}_5$ and

$$\Pr\left[\mathrm{Exp}^{(0)\text{-}\mathsf{inv}}_{(\overline{S},\overline{S}^{-1}),\mathcal{B}_5,\overline{f}}(\lambda) = 1\right] = \Pr\left[out_5 = 1 \,\middle|\, f = \overline{f}\right].$$

If, on the other hand, $\mathcal{B}_5$ plays the experiment $\mathrm{Exp}^{(1)\text{-}\mathsf{inv}}_{(\overline{S},\overline{S}^{-1}),\mathcal{B}_5,\overline{f}}(\lambda)$, he perfectly simulates game $\mathbf{G}_4$ and

$$\Pr\left[\mathrm{Exp}^{(1)\text{-}\mathsf{inv}}_{(\overline{S},\overline{S}^{-1}),\mathcal{B}_5,\overline{f}}(\lambda) = 1\right] = \Pr\left[out_4 = 1 \,\middle|\, f = \overline{f}\right].$$

Thus, the claim follows. □

Summing up, we have that $|\Pr[out_0 = 1] - \Pr[out_5 = 1]|$ is negligible.

LEMMA 17.8. $\Pr[out_5 = 1]$ *is overwhelming.*

Proof of Lemma 17.8. Since $\mathcal{F}$ is a GEOWF, we have that

$$\Pr_{\mathbf{G}_5}\left[\mathcal{R}_f^{\mathcal{F}}(y, x') = 1 \vee y \notin \mathit{image}(f)\right]$$

is overwhelming. Using a union bound, we get

$$\begin{aligned}\Pr_{\mathbf{G}_5}&\left[\mathcal{R}_f^{\mathcal{F}}(y, x') = 1 \vee y \notin \mathit{image}(f)\right] \\ &\leqslant \Pr_{\mathbf{G}_5}\left[\mathcal{R}_f^{\mathcal{F}}(y, x') = 1\right] + \Pr_{\mathbf{G}_5}\left[y \notin \mathit{image}(f)\right].\end{aligned}$$

In the following, we prove that $\Pr_{\mathbf{G}_5}[y \notin \mathit{image}(f)]$ is negligible. Consequently, $\Pr_{\mathbf{G}_5}\left[\mathcal{R}_f^{\mathcal{F}}(y, x') = 1\right] = \Pr[\mathit{out}_5 = 1]$ is overwhelming.

By construction, S on input of f always produces values $y \in \mathit{image}(f)$. If $\Pr_{\mathbf{G}_5}\left[y \notin \mathit{image}(f)\right] = \Pr\left[f \leftarrow \mathsf{Gen}(1^\lambda), y \leftarrow \overline{S}(f) \colon y \notin \mathit{image}(f)\right]$ is non-negligible, then an unbounded adversary can distinguish between outputs of $S(f)$ and outputs of $\overline{S}(f)$ simply by testing all possible pre-images. Let $\overline{\mathcal{B}}$ be the adversary who outputs 1 if and only if $y \notin \mathit{image}(f)$. Hence,

$$\Pr_{\mathbf{G}_5}\left[y \notin \mathit{image}(f)\right] = \sum_{\overline{f} \in \mathit{supp}(\mathsf{Gen}(1^\lambda))} \Pr[f = \overline{f}] \cdot \mathrm{Adv}^{\mathsf{close}}_{(\overline{S}, \overline{S}^{-1}), \overline{\mathcal{B}}, \overline{f}}(\lambda).$$

Since for all $f \in \mathit{supp}(\mathsf{Gen}(1^\lambda))$, $\mathrm{Adv}^{\mathsf{close}}_{(\overline{S}, \overline{S}^{-1}), \overline{\mathcal{B}}, \overline{f}}(\lambda)$ is negligible, $\Pr_{\mathbf{G}_5}\left[y \notin \mathit{image}(f)\right]$ is negligible. □

This concludes the proof of Theorem 17.7. □

From Theorem 17.7 together with Theorem 17.6, we obtain the following corollary.

COROLLARY 17.6. *Assuming the learning with errors problem is subexponentially hard, then* $\mathsf{PREH}^{\mathsf{rand}}_{\equiv_s}$ *is false.*

The above proof extends to $\mathsf{cPREH}^{\mathsf{rand}}_{\equiv_s}$ at the cost of assuming privately verifiable GEOWFs *with unbounded auxiliary information*. We omit the details here since no instantiation of this variant of GEOWFs is known and refer the reader to the proof of Theorem 17.10 for an example how to extend this strategy to the CRS case.

THEOREM 17.8. *If privately verifiable generalized extractable one-way functions with unbounded auxiliary information exist, then* $\mathsf{cPREH}^{\mathsf{rand}}_{\equiv_s}$ *is false.*

17.2.3 Computational Guarantees and EOWFs with Common Auxiliary Information

Assuming a pseudorandom encoding scheme providing computational guarantees, the above proof strategy must be adapted since the adversary actually uses the alternative sampler of a sampler which outputs an image in the range of the EOWF. Computational indistinguishability does not suffice to force the alternative sampler to output an image in the range of the EOWF. Hence, for the purpose to force the alternative sampler to output an image of a given extractable one-way function, we follow the lines of [IKOS10] and require the original sampler to additionally provide a non-interactive zero-knowledge proof certifying that the provided image is in the range of the given EOWF.

By this strategy [IKOS10] prove that $\mathsf{PREH}^{\mathsf{rand}}_{\approx_c}$ in conjunction with NIZK proof systems conflicts with EOWFs (without auxiliary input).

THEOREM 17.9 ([IKOS10]). *If extractable one-way functions without auxiliary information and NIZK proof systems for $\mathcal{NP}$ exist, then $\mathsf{PREH}^{\mathsf{rand}}_{\approx_c}$ is false.*

This result can be extended to $\mathsf{cPREH}^{\mathsf{rand}}_{\approx_c}$ at the cost of assuming EOWFs with unbounded common auxiliary inputs.

THEOREM 17.10. *If there exist extractable one-way functions with unbounded common auxiliary information and NIZK proof systems for $\mathcal{NP}$, then $\mathsf{cPREH}^{\mathsf{rand}}_{\approx_c}$ is false.*

Proof. We adapt the proof from [IKOS10]. Let $\mathcal{F}$ be an extractable one-way function with common auxiliary information. Let

$$L_\lambda := \left\{ (f, y) \in \mathit{supp}(\mathsf{Gen}(1^\lambda)) \times \{0,1\}^{m(\lambda)} \;\middle|\; \begin{array}{c} \exists x \in \{0,1\}^{n(\lambda)} \text{ s.t.} \\ f(x) = y \end{array} \right\}.$$

Let $(\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ be a NIZK proof system for L_λ such that Setup produces uniform random strings from $\{0,1\}^{n_\Pi(\lambda)}$.

$\mathsf{cPREH}^{\mathsf{rand}}_{\approx_c}$ implies that for the PPT algorithm S (see Figure 17.4), there exist a PPT algorithm Setup_S, an alternative sampler $\overline{S}$ and a corresponding inverse sampler $\overline{S}^{-1}$ satisfying closeness and invertibility as in Definition 16.3.

$S(f, \sigma)$

$x \leftarrow \{0,1\}^{n(\lambda)}$

$y := f(x)$

$\pi \leftarrow \mathsf{Prove}(\sigma, (f, y), x)$

return (y, π)

(a) Sampler mapping a random pre-image with f and proving membership in the range of f.

$\mathcal{X}(f, z =: crs; \sigma \,\|\, r_{\overline{S}})$

$(y, \pi) \leftarrow \overline{S}(crs, (f, \sigma); r_{\overline{S}})$

return y

(b) An adversary producing an image of f.

$\mathcal{A}(1^\lambda, f, y)$

$crs \leftarrow \mathsf{Setup}_S(1^\lambda)$

$(\sigma, \tau_\Pi) \leftarrow \mathsf{Sim}_0(1^\lambda)$

$\pi \leftarrow \mathsf{Sim}_1(\tau_\Pi, (f, y))$

$r'_{\overline{S}} \leftarrow \overline{S}^{-1}(crs, (f, \sigma), (y, \pi))$

$r'_{\mathcal{X}} := \sigma \,\|\, r'_{\overline{S}}$

$x' \leftarrow K_{\mathcal{X}}(f, crs, r'_{\mathcal{X}})$

return x'

(c) An adversary breaking one-wayness.

Figure 17.4: Description of the sampler S, of the adversary $\mathcal{X}$ against extractability of the EOWF and of the adversary $\mathcal{A}$ against one-wayness of the EOWF. Note that depending on the definition of one-wayness, the CRS *crs* could also be auxiliary input to $\mathcal{A}$.

The common auxiliary input is necessary to give the adversary $\mathcal{X}$ and the extractor $\mathcal{E}_\mathcal{X}$ access to the same common reference string. Further, it is crucial

to assume *unbounded* auxiliary input since the adversary $\mathcal{X}$ can be considered to be a universal adversary who simply executes the code which is contained in the auxiliary input. Since the size of the adversary cannot be bounded in advance, neither can the size of the auxiliary input.

Since $\mathcal{F}$ is an extractable one-way function with common auxiliary information, we have that for the algorithm $\mathcal{X}$ (see Figure 17.4b), there exists an extractor $\mathcal{E}_{\mathcal{X}}$ such that for every polynomial b and every $crs \in \{0,1\}^{b(\lambda)}$ (hence, in particular, for every $z := crs$ produced by $\mathsf{Setup}_S(1^\lambda)$),

$$\Pr\left[\mathrm{Exp}^{\text{ext-aux}}_{\mathcal{F},\mathcal{X},\mathcal{E}_{\mathcal{X}},z}(\lambda) = 1\right]$$

is overwhelming.

We prove that adversary $\mathcal{A}$ given in Figure 17.4c has an overwhelming probability to break the one-wayness of $\mathcal{F}$. We proceed over a sequence of hybrids, see Figures 17.5 and 17.6.

$\mathbf{G}_0$

$f \leftarrow \mathsf{Gen}(1^\lambda)$
$x \leftarrow \{0,1\}^{n(\lambda)}$
$y := f(x)$
$x' \leftarrow \mathcal{A}(f, y)$
return $f(x') = y$

(a) Description of $\mathbf{G}_0$.

$\mathbf{G}_1$

$f \leftarrow \mathsf{Gen}(1^\lambda)$
$x \leftarrow \{0,1\}^{n(\lambda)}$
$y := f(x)$
$crs \leftarrow \mathsf{Setup}_S(1^\lambda)$
$(\sigma, \tau_\Pi) \leftarrow \mathsf{Sim}_0(1^\lambda)$
$\pi \leftarrow \mathsf{Sim}_1(\tau_\Pi, (f, y))$
$r'_S \leftarrow \overline{S}^{-1}(crs, (f, \sigma), (y, \pi))$
$r'_{\mathcal{X}} := \sigma \| r'_S$
$x' \leftarrow \mathcal{E}_{\mathcal{X}}(f, crs, r'_{\mathcal{X}})$
return $f(x') = y$

(b) Description of $\mathbf{G}_1$.

$\mathbf{G}_2$

$f \leftarrow \mathsf{Gen}(1^\lambda)$
$x \leftarrow \{0,1\}^{n(\lambda)}$
$y := f(x)$
$crs \leftarrow \mathsf{Setup}_S(1^\lambda)$
$\sigma \leftarrow \{0,1\}^{n_\Pi(\lambda)}$
$\pi \leftarrow \mathsf{Prove}(crs, (f, y), x)$
$r'_S \leftarrow \overline{S}^{-1}(crs, (f, \sigma), (y, \pi))$
$r'_{\mathcal{X}} := \sigma \| r'_S$
$x' \leftarrow \mathcal{E}_{\mathcal{X}}(f, crs, r'_{\mathcal{X}})$
return $f(x') = y$

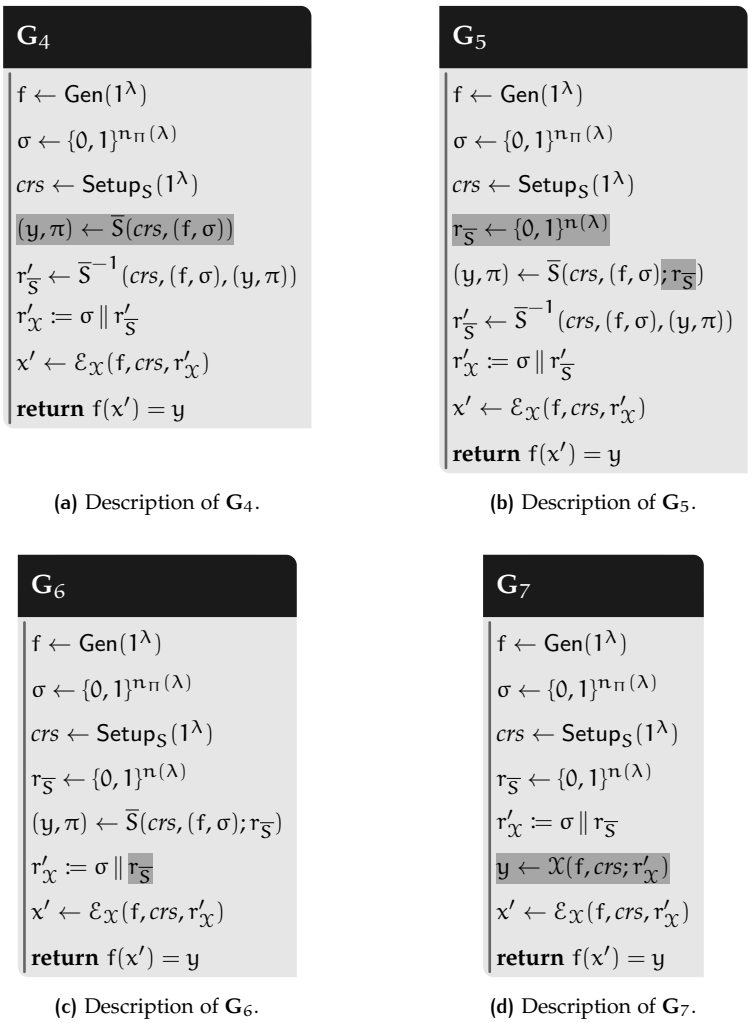
(c) Description of $\mathbf{G}_2$.

$\mathbf{G}_3$

$f \leftarrow \mathsf{Gen}(1^\lambda)$
$\sigma \leftarrow \{0,1\}^{n_\Pi(\lambda)}$
$crs \leftarrow \mathsf{Setup}_S(1^\lambda)$
$(y, \pi) \leftarrow S(f, \sigma)$
$r'_S \leftarrow \overline{S}^{-1}(crs, (f, \sigma), (y, \pi))$
$r'_{\mathcal{X}} := \sigma \| r'_S$
$x' \leftarrow \mathcal{E}_{\mathcal{X}}(f, crs, r'_{\mathcal{X}})$
return $f(x') = y$

(d) Description of $\mathbf{G}_3$.

Figure 17.5: Hybrids used in the proof of Theorem 17.10.



(a) Description of $\mathbf{G}_4$.

(b) Description of $\mathbf{G}_5$.

(c) Description of $\mathbf{G}_6$.

(d) Description of $\mathbf{G}_7$.

Figure 17.6: Hybrids used in the proof of Theorem 17.10.

GAME $\mathbf{G}_0$. $\mathbf{G}_0$ corresponds to $\mathrm{Exp}^{\mathsf{ow}}_{\mathcal{F},\mathcal{A}}(\lambda)$.

GAME $\mathbf{G}_1$. $\mathbf{G}_1$ is identical to $\mathbf{G}_0$ except that the implementation of $\mathcal{A}$ is explicit in $\mathbf{G}_1$. Hence, $\Pr[out_0 = 1] = \Pr[out_1 = 1]$.

GAME $\mathbf{G}_2$. $\mathbf{G}_2$ is identical to $\mathbf{G}_1$ except that in $\mathbf{G}_1$ σ and π are simulated using $(\mathsf{Sim}_0, \mathsf{Sim}_1)$, whereas in $\mathbf{G}_2$, σ is sampled uniformly at random from $\{0,1\}^{n_\Pi(\lambda)}$ and π is produced via $\mathsf{Prove}(\sigma, (f, y), x)$.

CLAIM 17.3. There exists a PPT adversary $\mathcal{B}_2$, such that $|\Pr[out_2 = 1] - \Pr[out_1 = 1]| \leqslant \mathrm{Adv}^{\mathsf{zk}}_{\mathsf{NIZK},\mathsf{Sim},\mathcal{B}_2}(\lambda)$.

Proof. Construct adversary $\mathcal{B}_2$ against the zero-knowledge property of NIZK. Initially, $\mathcal{B}_2$ receives σ as input, produces f, x, y and crs as in $\mathbf{G}_1$ and calls the prove oracle on input of the statement (f, y) together with a witness x. In return, $\mathcal{B}_2$ receives a proof π and proceeds as in $\mathbf{G}_1$ to produce $r'_{\overline{S}}$, r'_X and x'. Finally, $\mathcal{B}_2$ outputs 1 if $f(x') = y$ and 0 otherwise. If $\mathcal{B}_2$ receives a simulated CRS produced via $\mathsf{Sim}_0(1^\lambda)$ and a simulated proof produced via $\mathsf{Sim}_1(\sigma, \tau_\Pi, (f, y))$, $\mathcal{B}_2$ perfectly simulates $\mathbf{G}_1$. If, on the other hand, σ is uniformly random from $\{0,1\}^{n_\Pi(\lambda)}$ and the proof π

is produced by $\mathsf{Prove}(\sigma, (f, y), x)$, then $\mathcal{B}_2$ perfectly simulates $\mathbf{G}_2$. Hence, $|\Pr[out_2 = 1] - \Pr[out_1 = 1]| \leqslant \mathrm{Adv}^{\mathsf{zk}}_{\mathsf{NIZK},\mathsf{Sim},\mathcal{B}_2}(\lambda) = 1$. □

GAME $\mathbf{G}_3$. $\mathbf{G}_3$ and $\mathbf{G}_2$ are almost identical except for the difference that $\mathbf{G}_3$ does not produce y and π explicitly but uses $S(f, \sigma)$ to produce them. This difference is only conceptual and $\Pr[out_2 = 1] = \Pr[out_3 = 1]$.

GAME $\mathbf{G}_4$. $\mathbf{G}_4$ is identical to $\mathbf{G}_3$ except that $\mathbf{G}_3$ samples (y, π) using $S(f)$ whereas $\mathbf{G}_4$ samples (y, π) using $\overline{S}(f)$.

CLAIM 17.4. There exists a PPT adversary $\mathcal{B}_4$, such that $|\Pr[out_4 = 1] - \Pr[out_3 = 1]| \leqslant \mathrm{Adv}^{\text{crs-close}}_{(\mathsf{Setup}_S, \overline{S}, \overline{S}^{-1}), \mathcal{B}_4}(\lambda)$.

Proof. Construct a PPT adversary $\mathcal{B}_4$ against (static) closeness. Initially, $\mathcal{B}_4$ produces (f, σ) as in $\mathbf{G}_3$ and outputs them. In the second phase, $\mathcal{B}_4$ receives crs and (y, π), where (y, π) has either been sampled using $S(f, \sigma)$ or using $\overline{S}(crs, (f, \sigma))$. Further, $\mathcal{B}_4$ proceeds as in $\mathbf{G}_3$ producing r'_S, r'_X and x'. Finally, $\mathcal{B}_4$ outputs $f(x') = y$. Hence,

$$\Pr\left[out_3 = 1\right] = \Pr\left[\mathrm{Exp}^{(0)\text{-crs-close}}_{(\mathsf{Setup}_S, \overline{S}, \overline{S}^{-1}), \mathcal{B}_4}(\lambda) = 1\right] \text{ and}$$
$$\Pr\left[out_4 = 1\right] = \Pr\left[\mathrm{Exp}^{(1)\text{-crs-close}}_{(\mathsf{Setup}_S, \overline{S}, \overline{S}^{-1}), \mathcal{B}_4}(\lambda) = 1\right].$$

□

GAME $\mathbf{G}_5$. $\mathbf{G}_5$ is identical to $\mathbf{G}_4$ except that the random coins $r_{\overline{S}}$ used for $\overline{S}$ are made explicit. This difference is only conceptual and $\Pr[out_4 = 1] = \Pr[out_5 = 1]$.

GAME $\mathbf{G}_6$. $\mathbf{G}_6$ uses $r_{\overline{S}}$ for r'_X instead of the inverse sampled $r'_{\overline{S}}$. Apart from this difference, $\mathbf{G}_6$ is identical to $\mathbf{G}_5$.

CLAIM 17.5. There exists a PPT adversary $\mathcal{B}_6$, such that $|\Pr[out_6 = 1] - \Pr[out_5 = 1]| \leqslant \mathrm{Adv}^{\text{crs-inv}}_{(\mathsf{Setup}_S, \overline{S}, \overline{S}^{-1}), \mathcal{B}_6}(\lambda)$.

Proof. Construct a PPT adversary $\mathcal{B}_6$ on (static) invertibility. Initially, $\mathcal{B}_6$ (statically) produces (f, σ) as in $\mathbf{G}_5$ and outputs them to the experiment. In the second phase, $\mathcal{B}_6$ receives $(crs, r^*, (y, \pi))$, where (y, π) has been sampled using $\overline{S}(crs, (f, \sigma))$ and r^* either is the actual randomness used for that sampling process, or the inverse sampled randomness produced via $\overline{S}^{-1}(crs, (f, \sigma), (y, \pi))$. Afterwards, $\mathcal{B}_6$ proceeds as in $\mathbf{G}_5$ producing $r'_X := \sigma \,\|\, r^*$ and x'. Finally, $\mathcal{B}_6$ outputs $f(x') = y$. Hence,

$$\Pr\left[out_5 = 1\right] = \Pr\left[\mathrm{Exp}^{(1)\text{-crs-inv}}_{(\mathsf{Setup}_S, \overline{S}, \overline{S}^{-1}), \mathcal{B}_6}(\lambda) = 1\right] \text{ and}$$
$$\Pr\left[out_6 = 1\right] = \Pr\left[\mathrm{Exp}^{(0)\text{-crs-inv}}_{(\mathsf{Setup}_S, \overline{S}, \overline{S}^{-1}), \mathcal{B}_6}(\lambda) = 1\right].$$

□

GAME $\mathbf{G}_7$. $\mathbf{G}_7$ is identical to $\mathbf{G}_6$ except for the conceptional difference that $\mathbf{G}_7$ computes y using $X(f, crs; r'_X)$ and does not produce the proof π. Since π is not used anymore, this difference is only conceptual and $\Pr[out_6 = 1] = \Pr[out_7 = 1]$.

Thus, we have that $|\Pr[out_0 = 1] - \Pr[out_7 = 1]|$ is negligible.

LEMMA 17.9. $\Pr[out_7 = 1]$ *is overwhelming.*

Proof of Lemma 17.9. Since $\mathcal{F}$ is an extractable one-way function, we have that

$$\begin{aligned}&\Pr_{\mathbf{G}_7}\left[f(x') = y \vee (f,y) \notin L_\lambda\right]\\ &\quad\leqslant \Pr_{\mathbf{G}_7}\left[f(x') = y\right] + \Pr_{\mathbf{G}_7}\left[(f,y) \notin L_\lambda\right]\end{aligned}$$

is overwhelming. In the following, we prove that $\Pr_{\mathbf{G}_7}\left[(f,y) \notin L_\lambda\right]$ is negligible. We proceed over a series of hybrids, see Figure 17.7.

$\mathbf{H}_0$

$f \leftarrow \mathsf{Gen}(1^\lambda)$

$\sigma \,\|\, r_{\overline{S}} = r_{\mathcal{X}} \leftarrow \{0,1\}^{p(\lambda)}$

$\quad = \{0,1\}^{n_\Pi(\lambda)+n(\lambda)}$

$crs \leftarrow \mathsf{Setup}_S(1^\lambda)$

$y = \mathcal{X}(f, crs; r_{\mathcal{X}})$

return $(f,y) \notin L_\lambda$

(a) Description of $\mathbf{H}_0$.

$\mathbf{H}_1$

$f \leftarrow \mathsf{Gen}(1^\lambda)$

$\sigma \leftarrow \{0,1\}^{n_\Pi(\lambda)}$

$crs \leftarrow \mathsf{Setup}_S(1^\lambda)$

$(y,\pi) = \overline{S}(crs, (f,\sigma))$

return $(f,y) \notin L_\lambda$

(b) Description of $\mathbf{H}_1$.

$\mathbf{H}_2$

$f \leftarrow \mathsf{Gen}(1^\lambda)$

$\sigma \leftarrow \{0,1\}^{n_\Pi(\lambda)}$

$crs \leftarrow \mathsf{Setup}_S(1^\lambda)$

$(y,\pi) = \overline{S}(crs, (f,\sigma))$

return $(f,y) \notin L_\lambda$

$\quad \wedge\, \mathsf{Verify}(\sigma, (f,y), \pi) = 0$

(c) Description of $\mathbf{H}_2$.

$\mathbf{H}_3$

$f \leftarrow \mathsf{Gen}(1^\lambda)$

$\sigma \leftarrow \{0,1\}^{n_\Pi(\lambda)}$

$crs \leftarrow \mathsf{Setup}_S(1^\lambda)$

$(y,\pi) = \overline{S}(crs, (f,\sigma))$

return $\mathsf{Verify}(\sigma, (f,y), \pi) = 0$

(d) Description of $\mathbf{H}_3$.

$\mathbf{H}_4$

$f \leftarrow \mathsf{Gen}(1^\lambda)$

$\sigma \leftarrow \{0,1\}^{n_\Pi(\lambda)}$

$crs \leftarrow \mathsf{Setup}_S(1^\lambda)$

$(y,\pi) = S(f,\sigma)$

return $\mathsf{Verify}(\sigma, (f,y), \pi) = 0$

(e) Description of $\mathbf{H}_4$.

Figure 17.7: Hybrids used in the proof of Lemma 17.9.

GAME $\mathbf{H}_0$. $\mathbf{H}_0$ is identical to $\mathbf{G}_7$ except that $\mathbf{H}_0$ returns $(f,y) \notin L_\lambda$. That is, $\Pr[out_{\mathbf{H}_0} = 1] = \Pr_{\mathbf{G}_7}\left[(f,y) \notin L_\lambda\right]$.

GAME $\mathbf{H}_1$. $\mathbf{H}_1$ is identical to $\mathbf{H}_0$ except for the conceptual difference, that the algorithm $\mathcal{X}$ is replaced with its implementation. Hence, $\Pr[out_{\mathbf{H}_0} = 1] = \Pr[out_{\mathbf{H}_1} = 1]$.

GAME $\mathbf{H}_2$. $\mathbf{H}_2$ is identical to $\mathbf{H}_1$ except that it returns 1 only if $(f, y) \notin L_\lambda$ **and** $\mathsf{Verify}(\sigma, (f, y), \pi) = 0$.

CLAIM 17.6. There exists a PPT adversary $\mathcal{B}_{\mathbf{H}.2}$ such that $|\Pr[out_{\mathbf{H}_2} = 1] - \Pr[out_{\mathbf{H}_1} = 1]| \leqslant \mathrm{Adv}^{\mathsf{sound}}_{\mathsf{NIZK},\mathcal{B}_{\mathbf{H}.2}}(\lambda)$.

Proof. The only possibility that the output of $\mathbf{H}_1$ and $\mathbf{H}_2$ differ is the case that $(f, y) \notin L_\lambda$ but $\mathsf{Verify}(\sigma, (f, y), \pi) = 1$ (assuming, that the output of Verify is binary). Hence,

$$\left|\Pr\left[out_{\mathbf{H}_2} = 1\right] - \Pr\left[out_{\mathbf{H}_1} = 1\right]\right| \leqslant \Pr_{\mathbf{H}_1}\left[\begin{array}{c}(f, y) \notin L_\lambda \wedge \\ \mathsf{Verify}(\sigma, (f, y), \pi) = 1\end{array}\right] \leqslant \mathrm{Adv}^{\mathsf{sound}}_{\mathsf{NIZK},\mathcal{B}_{\mathbf{H}.2}}(\lambda).$$

The second inequality follows from the statistical soundness of NIZK. More precisely, construct an (unbounded) adversary $\mathcal{B}_{\mathbf{H}.2}$ on statistical soundness of the NIZK proof system. On input of σ, $\mathcal{B}_{\mathbf{H}.2}$ produces f, crs, (y, π) as in $\mathbf{H}_1$ and outputs the statement (f, y) and the proof π. □

GAME $\mathbf{H}_3$. The distribution underlying $\mathbf{H}_2$ and $\mathbf{H}_3$ is identical. The only difference between these games is that the condition that $\mathbf{H}_3$ outputs 1 is less restrictive than the one of $\mathbf{H}_2$, hence, $\Pr[out_{\mathbf{H}_2} = 1] \leqslant \Pr[out_{\mathbf{H}_3} = 1]$.

Note that in contrast to the preceding games, the simulation of game $\mathbf{H}_3$ is efficient.

GAME $\mathbf{H}_4$. $\mathbf{H}_4$ is identical to $\mathbf{H}_3$ except that in $\mathbf{H}_3$, (y, π) is produced using the alternative sampler $\overline{S}$ and in $\mathbf{H}_4$, (y, π) is produced using the original sampler S.

CLAIM 17.7. There exists a PPT adversary $\mathcal{B}_{\mathbf{H}.4}$ such that $|\Pr[out_{\mathbf{H}_4} = 1] - \Pr[out_{\mathbf{H}_3} = 1]| \leqslant \mathrm{Adv}^{\mathsf{crs\text{-}close}}_{(\mathsf{Setup}_S, \overline{S}, \overline{S}^{-1}), \mathcal{B}_{\mathbf{H}.4}}(\lambda)$.

Proof. Construct a PPT adversary $\mathcal{B}_{\mathbf{H}.4}$ against static closeness. Initially, $\mathcal{B}_{\mathbf{H}.4}$ produces (f, σ) as in $\mathbf{H}_3$ and outputs it to the experiment. On input of $(crs, \overline{y} := (y, \pi))$, where $\overline{y}$ is either sampled via $S(f, \sigma)$ or via $\overline{S}(crs, (f, \sigma))$, $\mathcal{B}_{\mathbf{H}.4}$ outputs $\mathsf{Verify}(\sigma, (f, y), \pi) = 0$. We have that

$$\Pr\left[out_{\mathbf{H}_3} = 1\right] = \Pr\left[\mathrm{Exp}^{(1)\text{-}\mathsf{crs\text{-}close}}_{(\mathsf{Setup}_S, \overline{S}, \overline{S}^{-1}), \mathcal{B}_{\mathbf{H}.4}}(\lambda) = 1\right] \text{ and}$$
$$\Pr\left[out_{\mathbf{H}_4} = 1\right] = \Pr\left[\mathrm{Exp}^{(0)\text{-}\mathsf{crs\text{-}close}}_{(\mathsf{Setup}_S, \overline{S}, \overline{S}^{-1}), \mathcal{B}_{\mathbf{H}.4}}(\lambda) = 1\right].$$

□

Finally, due to the definition of S and the completeness of the NIZK proof system NIZK,

$$\Pr\left[out_{\mathbf{H}_4} = 1\right] = 0.$$

This proves that

$$\Pr\left[out_{\mathbf{H}_0} = 1\right]$$

is negligible concluding the proof of Lemma 17.9. □

This concludes the proof of Theorem 17.10.

□

By an easy modification of the above proof, we can refute $\mathsf{cPREH}^{\mathsf{rand}}_{\approx_c}$ with universal setup.

THEOREM 17.11. *If there exist extractable one-way functions with unbounded common auxiliary information and NIZK proof systems for $\mathcal{NP}$, then $\mathsf{cPREH}^{\mathsf{rand}}_{\approx_c}$ with universal setup is false.*

ON THE NEED FOR UNBOUNDED COMMON AUXILIARY INPUT. We stress that assuming *unbounded* common auxiliary input is necessary since the adversary $\mathcal{X}$ can be considered the universal adversary. On a technical level, the common auxiliary input which leads to a contradiction is an output of the universal setup algorithm Setup on input of a sufficiently large bound B on the supported circuit size, or the "special" setup algorithm Setup_S. This bound B as well as the output size of Setup_S depend on the sampler size S and, hence, on the size of the adversary $\mathcal{X}$.

DISCUSSION. We stress that we are able to prove that EOWFs with unbounded common auxiliary input (in conjunction with NIZK proof systems) implies that $\mathsf{cPREH}^{\mathsf{rand}}_{\approx_c}$ is false. Furthermore, due to [DKR15] and Theorem 16.1 (or alternatively Theorems 17.13 and 17.14 in Section 17.3), $\mathsf{cPREH}^{\mathsf{rand}}_{\approx_c}$ can be instantiated from indistinguishability obfuscation and one-way functions. We restate a theorem from [BCPR16].

THEOREM 17.12 ([BCPR16]). *Assuming indistinguishability obfuscation for all circuits, neither EOWFs nor GEOWFs exist with respect to unbounded common auxiliary information.*

Thus, the above is not a contradiction because due to Theorem 17.12, indistinguishability obfuscation for all polynomial sized circuits does not exist assuming EOWFs with unbounded common auxiliary input.

COMMON BUT BENIGN AUXILIARY INFORMATION. The definition due to [BCPR16] of (G)EOWFs with common auxiliary input requires that extractability holds for all common auxiliary inputs and hence also for a *worst-case* choice of common auxiliary input. This requirement can be weakened such that the common auxiliary input is drawn from some specific distribution. This distribution is called *benign* if it is unlikely that a common auxiliary input sampled according to this distribution encodes a malicious obfuscation[65]. In particular, the uniform distribution over $\{0,1\}^{b(\lambda)}$ is conjectured to be benign. This notion of (G)EOWFs with common but benign auxiliary information does not contradict IO.

However, by an easy modification of the proof of Theorem 17.10, the existence of (G)EOWFs with common but benign auxiliary input (drawn uniformly at random from $\{0,1\}^{b(\lambda)}$) contradicts $\mathsf{cPREH}^{\mathsf{rand}}_{\approx_c}$ with common *random* string.

COROLLARY 17.7. *If there exist extractable one-way functions with common* but benign *auxiliary information, particularly for auxiliary inputs drawn uniformly at random from $\{0,1\}^{b(\lambda)}$, and NIZK proof systems for $\mathcal{NP}$, then $\mathsf{cPREH}^{\mathsf{rand}}_{\approx_c}$, where the setup algorithm produces uniform random strings, is false.*

[65] *By malicious obfuscation we mean an obfuscated circuit which renders extraction from an adversary who simply executes the obfuscated circuit on the given key infeasible, see [BCPR16].*

KEY-LESS EOWFS. Definitions 2.20, 2.22 and 17.10 can be defined for *key-less* function ensembles, where the key space $supp(\mathsf{Gen}(1^\lambda))$ contains exactly one element (for every λ). This poses much stronger requirements on one-wayness and extractability since these properties are bound to be met for one fixed key as opposed to the keyed variants.

As observed in [IKOS10], Theorems 17.7 and 17.9 can be adapted to the *weak* variants of $\mathsf{PREH}^{\mathsf{rand}}_{\approx_c}$ and $\mathsf{PREH}^{\mathsf{rand}}_{\equiv_s}$, respectively.[66] However, key-less (G)EOWFs with unbounded auxiliary input are impossible since the adversary might get a random image of the (fixed) (G)EOWF as auxiliary input. An extractor given the output of the adversary together with his random tape and the same auxiliary input must break one-wayness in order to produce a pre-image. Hence, Theorem 17.10 cannot be adapted to refute weak $\mathsf{cPREH}^{\mathsf{rand}}_{\approx_c}$.

[66] *For weak* $\mathsf{PREH}^{\mathsf{rand}}_{\approx_c}$*, the NIZK proof system must be replaced by a NIWI proof system (without CRS) and therefore the sampler* S *samples two images of the EOWF and a NIWI proof using one of the pre-images as witness.*

17.3 STATIC PSEUDORANDOM ENCODINGS FROM INDISTINGUISHABILITY OBFUSCATION

(Static) $\mathsf{cISH}^{\mathsf{rand}}_{\approx_c}$ (and hence $\mathsf{cPREH}^{\mathsf{rand}}_{\approx_c}$) is implied by the existence of an explainability compiler [DKR15] which can be built from indistinguishability obfuscation and one-way functions. Using the ideas from [SW14; DKR15], we obtain perfectly correct $\mathsf{cPREH}^{\mathsf{rand}}_{\approx_c}$ with and without universal setup. Note that perfect correctness does not follow from Theorem 16.1. Let S be a PPT algorithm and let $U_B(C, x; r)$ be the universal circuit that accepts any circuit C which can be represented with B bits and evaluates C on input x and randomness r.

Let B be an upper bound on the bitlength which is necessary to describe a sampler. Let $\ell_{in} = |m|$ be an upper bound on the bitlength of the inputs, ℓ_{out} be an upper bound on bitlength of the outputs and ℓ_r be an upper bound on the bitlength of random tape of such samplers. Let PRG be a PRG that maps $\{0,1\}^\lambda$ to $\{0,1\}^{2\lambda}$. Let $|u_1| = \ell_1 = 2B + 2\ell_{in} + 2\ell_{out} + 5\lambda$ and $|u_2| = \ell_2 = B + \ell_{in} + \ell_{out} + 2\lambda$.

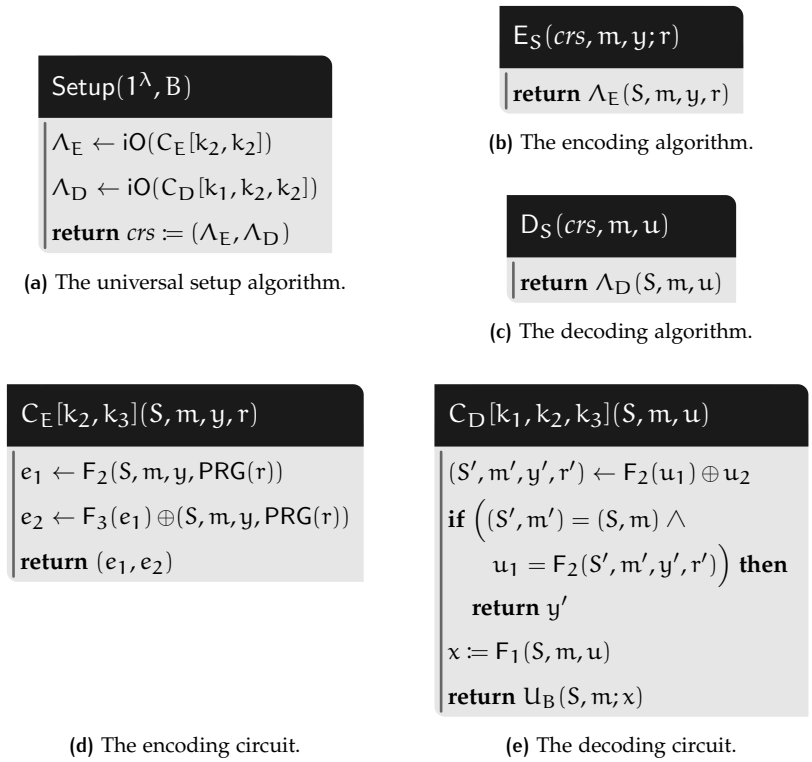
THEOREM 17.13. *Let* iO *be a perfectly correct indistinguishability obfuscator,* F_1, $\mathsf{F}_2, \mathsf{F}_3$ *be puncturable PRFs satisfying the following additional properties*

- F_1 *is extracting when the input min-entropy is greater than* $\ell_r + 2(\lambda+1) + 2$ *with error less than* $2^{-(\lambda+1)}$ *and has input length* $\ell_1 + \ell_2 + \ell_{in} + B$ *and output length* ℓ_r *(such a PRF exists from one-way functions since* $\ell_1 + \ell_2 + \ell_{in} + B \geqslant \ell_r + 2(\lambda+1) + 2$*),*
- F_2 *is statistically injective and has input length* $B + \ell_{in} + \ell_{out} + 2\lambda$ *and output length* ℓ_1 *(such a PRF exists from one-way functions since* $\ell_1 \geqslant 2(B + \ell_{in} + \ell_{out} + 2\lambda) + \lambda$*),*
- F_3 *has input length* ℓ_1 *and output length* ℓ_2*.*

Then, perfectly correct $\mathsf{cPREH}^{\mathsf{rand}}_{\approx_c}$ *with universal setup is true.*

Proof. We prove that $(\mathsf{Setup}, \mathsf{E}_S, \mathsf{D}_S)$ defined in Figure 17.8 satisfies perfect correctness and pseudorandomness.

PERFECT CORRECTNESS. Let $crs =: (\Lambda_E, \Lambda_D)$ be parameters in the support of $\mathsf{Setup}(1^\lambda, B)$. Let S be a PPT algorithm represented as a polynomial sized



(a) The universal setup algorithm.

(b) The encoding algorithm.

(c) The decoding algorithm.

(d) The encoding circuit.

(e) The decoding circuit.

Figure 17.8: Instantiation of perfectly correct $\mathsf{cPREH}^{\mathsf{rand}}_{\approx_c}$ due to [SW14; DKR15] with universal setup.

circuit, $m \in L$ be an input for S and let $y \in \mathit{supp}(S(m))$. Due to perfect correctness of iO we have

$$\begin{aligned} & D_S\left(crs, m, E_S\left(crs, m, y; r\right)\right) \\ = \; & C_D\left(S, m, C_E\left(S, m, y, r\right)\right) \\ = \; & C_D\Big(S, m, \big(\underbrace{F_2(S, m, y, \mathsf{PRG}(r))}_{=e_1}, \underbrace{F_3(e_1) \oplus (S, m, y, \mathsf{PRG}(r))}_{=e_2}\big)\Big) = y. \end{aligned}$$

PSEUDORANDOMNESS. The proof can be divided into two main steps. In Figure 17.9, we define the corresponding hybrids.

GAME $\mathbf{G}_0$. $\mathbf{G}_0$ corresponds to $\mathrm{Exp}^{(b)\text{-crs-pre}}_{(\mathsf{Setup},E_S,D_S),\mathcal{A}}(\lambda)$.

GAME $\mathbf{G}_1$. $\mathbf{G}_1$ is identical to $\mathbf{G}_0$ except that x^* (i. e., the randomness used by S) is produced differently. In $\mathbf{G}_0$, x^* is sampled uniformly at random, whereas in $\mathbf{G}_1$, x^* produced is via $F_1(S^*, m^*, u^*)$ (for uniformly random u^*).

CLAIM 17.8. For all (potentially unbounded) adversaries $\mathcal{A}$, $|\Pr[\mathit{out}_1 = 1] - \Pr[\mathit{out}_0 = 1]|$ is negligible.

The proof works as the proof of IND-CPA security of the deniable encryption scheme from [SW14] or the proof of statistical functional equivalence from [DKR15].

GAME $\mathbf{G}_2$. $\mathbf{G}_2$ is identical to $\mathbf{G}_1$ except that $\mathcal{A}$ does not receive the "explained" randomness e^* but the uniformly sampled randomness u^* as input.

CLAIM 17.9. For all PPT adversaries $\mathcal{A}$, $|\Pr[out_2 = 1] - \Pr[out_1 = 1]|$ is negligible.

The proof works as the proof of explainability of the deniable encryption scheme from [SW14] or the proof of explainability from [DKR15].

□

$\mathbf{G}_0$

$m^* \leftarrow \mathcal{A}(1^\lambda)$
$k_i \leftarrow F_i.\mathsf{KGen}(1^\lambda)$ (for $i \in [3]$)
$r^* \leftarrow \{0,1\}^\lambda$
$x^* \leftarrow \{0,1\}^{\ell_r}$
$y^* \leftarrow S^*(m^*; x^*)$
$e_1^* \leftarrow F_2(S^*, m^*, y^*, \mathsf{PRG}(r^*))$
$e_2^* \leftarrow F_3(e_1^*) \oplus (S^*, m^*, y^*, \mathsf{PRG}(r^*))$
$\Lambda_E \leftarrow \mathsf{iO}(C_E[k_2, k_3])$
$\Lambda_D \leftarrow \mathsf{iO}(C_D[k_1, k_2, k_3])$
$crs := (\Lambda_E, \Lambda_D)$
return $\mathcal{A}(crs, e^*)$

(a) Description of $\mathbf{G}_0$.

$\mathbf{G}_1$

$m^* \leftarrow \mathcal{A}(1^\lambda)$
$k_i \leftarrow F_i.\mathsf{KGen}(1^\lambda)$ (for $i \in [3]$)
$r^* \leftarrow \{0,1\}^\lambda, u^* \leftarrow \{0,1\}^{\ell_1+\ell_2}$
$x^* \leftarrow F_1(S^*, m^*, u^*)$
$y^* \leftarrow S^*(m^*; x^*)$
$e_1^* \leftarrow F_2(S^*, m^*, y^*, \mathsf{PRG}(r^*))$
$e_2^* \leftarrow F_3(e_1^*) \oplus (S^*, m^*, y^*, \mathsf{PRG}(r^*))$
$\Lambda_E \leftarrow \mathsf{iO}(C_E[k_2, k_3])$
$\Lambda_D \leftarrow \mathsf{iO}(C_D[k_1, k_2, k_3])$
$crs := (\Lambda_E, \Lambda_D)$
return $\mathcal{A}(crs, e^*)$

(b) Description of $\mathbf{G}_1$.

$\mathbf{G}_2$

$m^* \leftarrow \mathcal{A}(1^\lambda)$
$k_i \leftarrow F_i.\mathsf{KGen}(1^\lambda)$ (for $i \in [3]$)
$r^* \leftarrow \{0,1\}^\lambda, u^* \leftarrow \{0,1\}^{\ell_1+\ell_2}$
$x^* \leftarrow F_1(S^*, m^*, u^*)$
$y^* \leftarrow S^*(m^*; x^*)$
$e_1^* \leftarrow F_2(S^*, m^*, y^*, \mathsf{PRG}(r^*))$
$e_2^* \leftarrow F_3(e_1^*) \oplus (S^*, m^*, y^*, \mathsf{PRG}(r^*))$
$\Lambda_E \leftarrow \mathsf{iO}(C_E[k_2, k_3])$
$\Lambda_D \leftarrow \mathsf{iO}(C_D[k_1, k_2, k_3])$
$crs := (\Lambda_E, \Lambda_D)$
return $\mathcal{A}(crs, u^*)$

(c) Description of $\mathbf{G}_2$.

Figure 17.9: Hybrids used in the proof of pseudorandomness for Theorem 17.13.

By allowing the setup algorithm Setup_S to depend on S and replacing the universal circuit in C_D with S, we obtain $\mathsf{cPREH}^{\mathsf{rand}}_{\approx_c}$ without universal setup.

Let S be a PPT sampler. Let $\ell_{\mathsf{in}} = |m|$ be an upper bound on the bitlength of the inputs, ℓ_{out} be an upper bound on bitlength of the outputs and ℓ_r be an upper bound on the bitlength of random tape of S. Let PRG be a PRG that maps $\{0,1\}^\lambda$ to $\{0,1\}^{2\lambda}$. Let $|u_1| = \ell_1 = 2\ell_{\mathsf{in}} + 2\ell_{\mathsf{out}} + 5\lambda + \ell_r$ and $|u_2| = \ell_2 = \ell_{\mathsf{in}} + \ell_{\mathsf{out}} + 2\lambda$.

THEOREM 17.14. *Let* iO *be a perfectly correct indistinguishability obfuscator,* F_1, F_2, F_3 *be puncturable PRFs satisfying the following additional properties*

- F_1 *is extracting when the input min-entropy is greater than* $\ell_r + 2(\lambda+1)+2$ *with error less than* $2^{-(\lambda+1)}$ *and has input length* $\ell_1+\ell_2+\ell_{in}$ *and output length* ℓ_r *(such a PRF exists from one-way functions since* $\ell_1+\ell_2+\ell_{in}+\ell_r \geqslant \ell_r+2(\lambda+1)+2$*),*
- F_2 *is statistically injective and has input length* $\ell_{in}+\ell_{out}+2\lambda$ *and output length* ℓ_1 *(such a PRF exists from one-way functions since* $\ell_1 \geqslant 2(\ell_{in}+\ell_{out}+2\lambda)+\lambda$*),*
- F_3 *has input length* ℓ_1 *and output length* ℓ_2*.*

Then, perfectly correct $\mathsf{cPREH}^{\mathsf{rand}}_{\approx_c}$ *is true.*

The proof of Theorem 17.14 proceeds identically to Theorem 17.13 with $(\mathsf{Setup}_S, \mathsf{E}_S, \mathsf{D}_S)$ as defined in Figure 17.10. The only differences are changed input and output lengths for the pseudorandom functions.

Theorems 17.13 and 17.14 together with Theorem 15.1 yield the following corollary.

COROLLARY 17.8. *Assuming polynomially secure IO for all circuits and one-way functions, then perfectly correct* $\mathsf{acPREH}^{\mathsf{rand}}_{\approx_c}$ *(with or without universal setup) is true.*

$\mathsf{Setup}'_S(1^\lambda)$

$\Lambda_E \leftarrow \mathsf{iO}(C'_E[S, k_2, k_2])$

$\Lambda_D \leftarrow \mathsf{iO}(C'_D[S, k_1, k_2, k_2])$

return $crs \coloneqq (\Lambda_E, \Lambda_D)$

(a) The setup algorithm for S.

$\mathsf{E}'_S(crs, m, y; r)$

return $\Lambda_E(m, y, r)$

(b) The encoding algorithm.

$\mathsf{D}'_S(crs, m, u)$

return $\Lambda_D(m, u)$

(c) The decoding algorithm.

$C'_E[S, k_2, k_3](m, y, r)$

$e_1 \leftarrow \mathsf{F}_2(m, y, \mathsf{PRG}(r))$

$e_2 \leftarrow \mathsf{F}_3(e_1) \oplus (m, y, \mathsf{PRG}(r))$

return (e_1, e_2)

(d) The encoding circuit.

$C'_D[S, k_1, k_2, k_3](m, u)$

$(m', y', r') \leftarrow \mathsf{F}_2(u_1) \oplus u_2$

if $\Big(m' = m \wedge u_1 = \mathsf{F}_2(m', y', r')\Big)$ **then**

  return y'

$x \coloneqq \mathsf{F}_1(m, u)$

return $S(m; x)$

(e) The decoding circuit.

Figure 17.10: Instantiation of perfectly correct $\mathsf{cPREH}^{\mathsf{rand}}_{\approx_c}$ due to [SW14; DKR15] without universal setup.

17.4 BOOTSTRAPPING PSEUDORANDOM ENCODINGS WITH A COMMON RANDOM STRING

Recall that $\mathsf{cPREH}^{\mathsf{rand}}_{\approx_c}$ where the setup algorithm produces uniform random strings (i. e., in the common random string model) conflicts with EOWFs with common but benign auxiliary input – an assumption which we believe to be true. However, the very nature of pseudorandom encodings allows to bootstrap pseudorandom encodings in the common random string model from pseudorandom encodings with arbitrary setup if this setup algorithm – interpreted as a sampler – can be pseudorandomly encoded such that the corresponding setup algorithm produces common random strings. Note that this sampler does not expect an input. Via this bootstrapping, we are able to refute even the *weak* pseudorandom encoding hypothesis with common random string (URC).

THEOREM 17.15. *Assume (i)* $\mathsf{cPREH}^{\mathsf{rand}}_{\approx_c}$ *is true for all PPT samplers, i. e., for all PPT algorithms* S*, there exists a pseudorandom encoding scheme* $(\mathsf{Setup}'_S, \mathsf{E}'_S, \mathsf{D}'_S)$*, and (ii)* weak $\mathsf{cPREH}^{\mathsf{rand}}_{\approx_c}$ *is true for the class of PPT algorithms* Setup'_S *such that the corresponding setup algorithm* $\mathsf{Setup}''_{\mathsf{Setup}'_S}$ *produces uniform random strings. Then,* $\mathsf{cPREH}^{\mathsf{rand}}_{\approx_c}$ *with common random strings is true for all PPT samplers.*

Proof. Item i implies that for all PPT algorithms S, there exists a pseudorandom encoding scheme $(\mathsf{Setup}'_S, \mathsf{E}'_S, \mathsf{D}'_S)$. Further, Item ii guarantees the existence of a pseudorandom encoding scheme $(\mathsf{Setup}''_{\mathsf{Setup}'_S}, \mathsf{E}''_{\mathsf{Setup}'_S}, \mathsf{D}''_{\mathsf{Setup}'_S})$ such that $\mathsf{Setup}''_{\mathsf{Setup}'_S}$ produces uniform random strings. Let $\{0,1\}^{n''(\lambda)}$ be the range of $\mathsf{E}''_{\mathsf{Setup}'_S}$ (note that $\{0,1\}^{n''(\lambda)}$ is allowed to depend on the sampler Setup'_S). For notational convenience, we henceforth omit the dependency on the sampler Setup'_S.

Let S be some PPT algorithm. We define $(\mathsf{Setup}_S, \mathsf{E}_S, \mathsf{D}_S)$ corresponding to S as in Figure 17.11.

$\mathsf{Setup}_S(1^\lambda)$

$crs'' \leftarrow \mathsf{Setup}''(1^\lambda)$
$u'' \leftarrow \{0,1\}^{n''(\lambda)}$
$crs \coloneqq crs'' \| u''$
return crs

(a) Setup algorithm.

$\mathsf{E}_S(crs, m, y)$

parse $crs =: crs'' \| u''$
$crs' \leftarrow \mathsf{D}''(crs'', u'')$
$u \leftarrow \mathsf{E}'_S(crs', m, y)$
return u

(b) Encoding algorithm.

$\mathsf{D}_S(crs, m, u)$

parse $crs =: crs'' \| u''$
$crs' \leftarrow \mathsf{D}''(crs'', u'')$
$y \leftarrow \mathsf{D}'_S(crs', m, u)$
return y

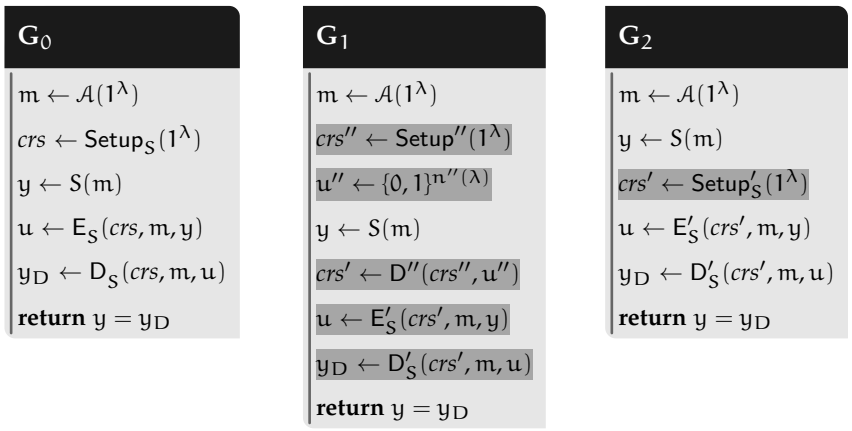
(c) Decoding algorithm.

Figure 17.11: Algorithms for $\mathsf{cPREH}^{\mathsf{rand}}_{\approx_c}$ with common *random* string.

CORRECTNESS. We use a sequence of hybrids, see Figure 17.12.

GAME $\mathbf{G}_0$. $\mathbf{G}_0$ encodes and decodes a sample y (using a statically chosen input m) using $(\mathsf{Setup}_S, \mathsf{E}_S, \mathsf{D}_S)$.

GAME $\mathbf{G}_1$. $\mathbf{G}_1$ is identical to $\mathbf{G}_0$ except that $\mathbf{G}_1$ uses the implementations of Setup_S, E_S and D_S explicitly. This difference is only conceptual and $\Pr[out_0 = 1] = \Pr[out_1 = 1]$.



(a) Description of $\mathbf{G}_0$. **(b)** Description of $\mathbf{G}_1$. **(c)** Description of $\mathbf{G}_2$.

Figure 17.12: Hybrids used in the proof of correctness of Theorem 17.15.

GAME $\mathbf{G}_2$. $\mathbf{G}_2$ is identical to $\mathbf{G}_1$ except that crs' is sampled using $\mathsf{Setup}'_S(1^\lambda)$ instead of decoding u'' using $\mathsf{D}''(crs'', u'')$.

Due to Lemma 16.2, by correctness and pseudorandomness of $(\mathsf{Setup}'', \mathsf{E}'', \mathsf{D}'')$, the two distributions

$$\left\{crs'' \leftarrow \mathsf{Setup}''(1^\lambda), crs' \leftarrow \mathsf{Setup}'_S(1^\lambda) : (crs'', crs')\right\} \text{ and}$$
$$\left\{crs'' \leftarrow \mathsf{Setup}''(1^\lambda), crs' \leftarrow \mathsf{D}''(crs'', U_{n''(\lambda)}) : (crs'', crs')\right\}$$

are computationally indistinguishable. Hence, $|\Pr[out_2 = 1] - \Pr[out_1 = 1]|$ is negligible.

Due to correctness of $(\mathsf{Setup}'_S, \mathsf{E}'_S, \mathsf{D}'_S)$, for all PPT adversaries $\mathcal{A}$,

$$\Pr\left[\begin{array}{ll} m & \leftarrow \mathcal{A}(1^\lambda) \\ crs' & \leftarrow \mathsf{Setup}'_S(1^\lambda) : \mathsf{D}'_S(crs', m, \mathsf{E}'_S(crs', m, y)) = y \\ y & \leftarrow S(m) \end{array}\right]$$

and, hence, $\Pr[out_2 = 1]$ are overwhelming.

Before we prove pseudorandomness, we introduce a technical lemma.

LEMMA 17.10. *Let S be a PPT sampler such that $\mathsf{cPREH}^{\mathsf{rand}}_{\approx_c}$ is true for S and let $(\mathsf{Setup}_S, \mathsf{E}_S, \mathsf{D}_S)$ be the corresponding algorithms (such that E_S has output length $n(\lambda)$). Then, for all PPT adversaries $\mathcal{A}$,*

$$\left|\Pr\left[\begin{array}{ll} m & \leftarrow \mathcal{A}(1^\lambda) \\ crs & \leftarrow \mathsf{Setup}_S(1^\lambda) \\ y & \leftarrow S(m) \\ u & \leftarrow \mathsf{E}_S(crs, m, y) \end{array} : \mathcal{A}(crs, m, y, u) = 1\right]\right.$$
$$\left. - \Pr\left[\begin{array}{ll} m & \leftarrow \mathcal{A}(1^\lambda) \\ crs & \leftarrow \mathsf{Setup}_S(1^\lambda) \\ u & \leftarrow \{0,1\}^{n(\lambda)} \\ y & \leftarrow \mathsf{D}_S(crs, m, u) \end{array} : \mathcal{A}(crs, m, y, u) = 1\right]\right|$$

is negligible.

Proof of Lemma 17.10. We proceed over a series of hybrids, see Figure 17.13.

GAME $\mathbf{H}_0$. $\mathbf{H}_0$ corresponds to the left-hand-side of the difference, where $\mathcal{A}$ receives a sample y from $S(m)$ and the corresponding encoded sample $u \leftarrow E_S(crs, m, y)$.

GAME $\mathbf{H}_1$. $\mathbf{H}_1$ is identical to $\mathbf{H}_0$ except that $\mathcal{A}$ receives the decoded sample y_D instead of the actual sample y_S. If $y_S = y_D$ both games behave identically, hence, by correctness, $|\Pr[out_{\mathbf{H}_1} = 1] - \Pr[out_{\mathbf{H}_0} = 1]| \leqslant \epsilon^{\text{c-dec-error}}_{(\mathsf{Setup}_S, \mathsf{E}_S, \mathsf{D}_S), \mathcal{A}}(\lambda)$.

GAME $\mathbf{H}_2$. $\mathbf{H}_2$ is identical to $\mathbf{H}_1$ except that u is not produced as an encoding of y, but sampled uniformly at random from $\{0,1\}^{n(\lambda)}$. This game hop is justified by pseudorandomness $|\Pr[out_{\mathbf{H}_2} = 1] - \Pr[out_{\mathbf{H}_1} = 1]| \leqslant \mathrm{Adv}^{\text{crs-pre}}_{(\mathsf{Setup}_S, \mathsf{E}_S, \mathsf{D}_S), \mathcal{A}}(\lambda)$.

$\mathbf{H}_0$

$m \leftarrow \mathcal{A}(1^\lambda)$
$crs \leftarrow \mathsf{Setup}_S(1^\lambda)$
$y_S \leftarrow S(m)$
$u \leftarrow \mathsf{E}_S(crs, m, y)$
return $\mathcal{A}(crs, m, y_S, u)$

(a) Description of $\mathbf{H}_0$.

$\mathbf{H}_1$

$m \leftarrow \mathcal{A}(1^\lambda)$
$crs \leftarrow \mathsf{Setup}_S(1^\lambda)$
$y_S \leftarrow S(m)$
$u \leftarrow \mathsf{E}_S(crs, m, y)$
$y_D \leftarrow \mathsf{D}_S(crs, m, u)$
return $\mathcal{A}(crs, m, y_D, u)$

(b) Description of $\mathbf{H}_1$.

$\mathbf{H}_2$

$m \leftarrow \mathcal{A}(1^\lambda)$
$crs \leftarrow \mathsf{Setup}_S(1^\lambda)$
$u \leftarrow \{0,1\}^{n(\lambda)}$
$y_D \leftarrow \mathsf{D}_S(crs, m, u)$
return $\mathcal{A}(crs, m, y_D, u)$

(c) Description of $\mathbf{H}_2$.

Figure 17.13: Hybrids used in the proof of Lemma 17.10.

This concludes the proof of Lemma 17.10. □

PSEUDORANDOMNESS. To prove pseudorandomness, we proceed over a sequence of hybrids, see Figure 17.14.

$\mathbf{G}_0$

$m \leftarrow \mathcal{A}(1^\lambda)$
$crs'' \leftarrow \mathsf{Setup}''(1^\lambda)$
$u'' \leftarrow \{0,1\}^{n''(\lambda)}$
$crs' \leftarrow \mathsf{D}''(crs'', u'')$
$crs \coloneqq crs'' \| u''$
$y \leftarrow S(m)$
$u \leftarrow \mathsf{E}'_S(crs', m, y)$
return $\mathcal{A}(crs, u)$

(a) Description of $\mathbf{G}_0$.

$\mathbf{G}_1$

$m \leftarrow \mathcal{A}(1^\lambda)$
$crs'' \leftarrow \mathsf{Setup}''(1^\lambda)$
$crs' \leftarrow \mathsf{Setup}'_S(1^\lambda)$
$u'' \leftarrow \mathsf{E}''(crs'', crs')$
$crs \coloneqq crs'' \| u''$
$y \leftarrow S(m)$
$u \leftarrow \mathsf{E}'_S(crs', m, y)$
return $\mathcal{A}(crs, u)$

(b) Description of $\mathbf{G}_1$.

$\mathbf{G}_2$

$m \leftarrow \mathcal{A}(1^\lambda)$
$crs'' \leftarrow \mathsf{Setup}''(1^\lambda)$
$crs' \leftarrow \mathsf{Setup}'_S(1^\lambda)$
$u'' \leftarrow \mathsf{E}''(crs'', crs')$
$crs \coloneqq crs'' \| u''$
$y \leftarrow S(m)$
$u \leftarrow \{0,1\}^{n'(\lambda)}$
return $\mathcal{A}(crs, u)$

(c) Description of $\mathbf{G}_2$.

Figure 17.14: Hybrids used in the proof of pseudorandomness of Theorem 17.15.

GAME $\mathbf{G}_0$. $\mathbf{G}_0$ corresponds to $\mathrm{Exp}^{(0)\text{-crs-pre}}_{(\mathsf{Setup}_S, \mathsf{E}_S, \mathsf{D}_S), \mathcal{A}}(\lambda)$.

GAME $\mathbf{G}_1$. $\mathbf{G}_1$ is identical to $\mathbf{G}_0$ except for the following difference. $\mathbf{G}_1$ produces crs' directly via $\mathsf{Setup}'_S(1^\lambda)$ and produces u'' via encoding crs'

with $\mathsf{E}''(crs'', crs')$. $\mathbf{G}_0$, on the other hand, samples u'' uniformly at random from $\{0,1\}^{n''(\lambda)}$ and produces crs'' via decoding u'' as $\mathsf{D}''(crs'', u'')$. Due to Lemma 17.10, $|\Pr[out_1 = 1] - \Pr[out_0 = 1]|$ is negligible.

GAME $\mathbf{G}_2$. $\mathbf{G}_2$ is identical to $\mathbf{G}_1$ except that $\mathbf{G}_2$ samples u uniformly at random from $\{0,1\}^{n'(\lambda)}$, whereas $\mathbf{G}_1$ produces u as $\mathsf{E}'_S(crs', m, y)$. For all PPT adversaries $\mathcal{A}$, $|\Pr[out_2 = 1] - \Pr[out_1 = 1]|$ is negligible due to pseudorandomness of $(\mathsf{Setup}'_S, \mathsf{E}'_S, \mathsf{D}'_S)$.

□

Combining Theorem 17.15 and Corollary 17.7, we obtain the following corollary refuting weak $\mathsf{cPREH}^{\mathsf{rand}}_{\approx_c}$ with common random string.

COROLLARY 17.9. *If there exist extractable one-way function family ensembles with common but benign auxiliary information and indistinguishability obfuscation for all circuits, then* weak $\mathsf{cPREH}^{\mathsf{rand}}_{\approx_c}$ *with common random string (URC) is false.*

REMARK 17.1. If weak $\mathsf{cPREH}^{\mathsf{rand}}_{\approx_c}$ was true such that the setup algorithms always produce uniform random strings, then the global common reference string model could be replaced with the global common random string model (or the non-programmable random oracle model) in several settings in cryptography.

18 APPLICATIONS OF PSEUDORANDOM ENCODINGS

In this chapter we describe further applications of pseudorandom encodings. This unifies several areas in cryptography. In Section 18.1, we show that adaptively secure pseudorandom encodings with setup are equivalent to fully adaptive multi-party computation in the global CRS model for all PPT functionalities (that includes non-adaptively-well-formed functionalities). This extends the results from [IKOS10] and clarifies a claim made in [DKR15]. In Section 18.2, we show that pseudorandom encodings yield honey encryption due to [JR14] for arbitrary message distributions, even such which admit inputs. In Section 18.3, we define a keyless version of steganography. That is, in contrast to symmetric-key or public-key steganography, parties do not need any secret information in order to covertly communicate with each other. In Sections 18.4 and 18.5, we analyze the relation of pseudorandom encodings and certain PKE variants which are known to imply variants of adaptive MPC.

18.1 ADAPTIVELY SECURE MULTI-PARTY COMPUTATION

Due to the equivalence of pseudorandom encodings and invertible sampling, we obtain an equivalence between pseudorandom encodings and fully adaptively secure multi-party computation. Due to [IKOS10], $\mathsf{PREH}^{\mathsf{rand}}_{\approx_c}$ in conjunction with some adaptively secure oblivious transfer protocol implies fully adaptively secure multi-party computation for all randomized functionalities in the plain model.

In Section 18.1.1, we introduce some preliminaries on adaptive multi-party computation, henceforth denoted AMPC. In Sections 18.1.2 and 18.1.3, we extend the results on the relation between the invertible sampling hypothesis and AMPC from [IKOS10] to the global CRS model. This is directly possible since, due to Theorem 15.1 and Corollary 16.1, we have an instantiation of $\mathsf{acISH}^{\mathsf{rand}}_{\approx_c}$.

18.1.1 Adaptive MPC

The definition of security of multi-party protocols follows the real/ideal model paradigm, [GMW87; Canoo]. A protocol Π is said to be secure if the output of the real execution of the protocol is indistinguishable from the output of an ideal computation, where a trusted third party exists. The behavior of this trusted third party is defined via a functionality $\mathcal{F}$.

Let $\mathcal{F}$ be a PPT functionality. Let x_i be the input to party P_i. We only consider semi-honest adversaries. Semi-honest adversaries are bound to follow the protocol specification while trying to obtain as much information

as possible. There are general techniques to transform protocols which are secure with respect to semi-honest adversaries to protocols which are secure with respect to malicious adversaries at the cost of a local CRS which is inherent for malicious security, [CLOS02]. In the following, we describe the stand-alone model for adaptive MPC on a very high level. We refer the reader to [Canoo; Lino9] for more details.

THE IDEAL EXECUTION. The ideal execution involves an ideal PPT adversary Sim (also called the simulator), a PPT environment $\mathcal{Z}$ and a trusted third party T. The ideal execution proceeds through several stages. In the *first corruption stage*, Sim adaptively decides to corrupt parties. When Sim corrupts a party, Sim learns that party's input and $\mathcal{Z}$ learns the identity of the corrupted party. In the *computation stage*, the uncorrupted and the corrupted parties send their inputs (which are determined by the environment) to T. The trusted party T evaluates the (possibly randomized) function $(z_1, \ldots, z_n) \leftarrow \mathcal{F}(x_1, \ldots, x_n)$ and sends the output to the respective parties. In the *second corruption stage*, Sim learns the outputs of the corrupted parties and again adaptively decides to corrupt a party. Upon corruption of a party, Sim learns that party's input *and output* and $\mathcal{Z}$ learns the corrupted party's identity. In the *output stage*, the uncorrupted parties output what they received from T to $\mathcal{Z}$, the corrupted parties output $\perp$ to $\mathcal{Z}$ and Sim outputs an arbitrary string to $\mathcal{Z}$. Finally, in the *post-execution corruption stage*, as long as the environment $\mathcal{Z}$ did not halt, $\mathcal{Z}$ sends corruption requests to Sim. Sim generates an arbitrary answer based on its view so far. Furthermore, Sim may additionally corrupt parties as in the second corruption stage.

Let $\mathcal{Z}^{\mathsf{Sim},\mathcal{F}}(1^\lambda)$ denote the output distribution of the environment $\mathcal{Z}$ when interacting as described above with the simulator Sim and parties $P_1, \ldots, P_n$ on inputs $x_1, \ldots, x_n$ chosen by $\mathcal{Z}$.

THE REAL EXECUTION. Initially, the environment $\mathcal{Z}$ (adaptively) chooses inputs $x_1, \ldots, x_n$ and sends each party P_i its input. Before the communication rounds start, the adversary $\mathcal{A}$ receives an initial message from $\mathcal{Z}$. While there exist uncorrupted parties which did not halt, $\mathcal{A}$ may adaptively decide to corrupt new parties. Upon corruption of P_i, $\mathcal{Z}$ learns the identity of P_i, $\mathcal{A}$ learns the input x_i, P_i's internal state (i. e., its random tape) and all messages received so far. From that time on, $\mathcal{A}$ is in control of the messages this party sends. Furthermore, if a corrupted party receives a message, $\mathcal{A}$ also learns that message. Additionally, $\mathcal{A}$ determines the order in which the uncorrupted parties are activated. If all parties are corrupted or all uncorrupted parties halted, each uncorrupted party and $\mathcal{A}$ produce outputs. The environment $\mathcal{Z}$ learns all of these outputs. Similarly to the post-execution corruption stage in the ideal execution, while $\mathcal{Z}$ did not halt, $\mathcal{Z}$ may instruct $\mathcal{A}$ to corrupt more parties or $\mathcal{A}$ may decide himself to corrupt more parties. Upon corruption, $\mathcal{A}$ learns P_i's input, randomness and all messages received so far and $\mathcal{Z}$ learns the corrupted party's identity. Additionally, $\mathcal{A}$ sends P_i's internal state to $\mathcal{Z}$.

Let $\mathcal{Z}^{\mathcal{A},\Pi}(1^\lambda)$ denote the output of the environment $\mathcal{Z}$ when interacting with the adversary $\mathcal{A}$ and parties $P_1, \ldots, P_n$ running the protocol Π on inputs $x_1, \ldots, x_n$ chosen by $\mathcal{Z}$.

DEFINITION 18.1 (Adaptive security in the stand-alone model, [Canoo]). We say a *protocol* Π *computes functionality* $\mathcal{F}$ *in the adaptive semi-honest model* if for

every PPT adversary $\mathcal{A}$, there exists a PPT simulator Sim, such that for all PPT environments $\mathcal{Z}$,

$$\left|\Pr\left[\mathcal{Z}^{\mathcal{A},\Pi}(1^\lambda)=1\right]-\Pr\left[\mathcal{Z}^{\mathsf{Sim},\mathcal{F}}(1^\lambda)=1\right]\right|$$

is negligible.

COMMON REFERENCE STRING MODEL. In the common reference string (CRS) model, all parties have access to a string which is honestly generated by a trusted third non-participating party. There are two widely used variants of the CRS model. In the *programmable* or *local* CRS model, the simulator may generate the CRS. This enables the simulator to sample the CRS along with corresponding trapdoors which results in an asymmetry between the simulator and the adversary facilitating simulation. However, as soon as two different protocols use the same programmable CRS, all security guarantees related to that CRS break down, see [CDPW07]. In the *non-programmable* or *global* CRS model, the simulator receives the CRS as input and, hence, has no additional power compared to the adversary. A global CRS can be made public and used by any number of protocols without compromising security.

Adaptive security in the global CRS model (with respect to some efficiently samplable CRS distribution Setup) is defined as Definition 18.1 with the difference that all parties including the environment, the adversary and the simulator receive the CRS as input. We stress that the environment may decide on the inputs for the parties adaptively after seeing the CRS.

THE UNIVERSAL COMPOSABILITY FRAMEWORK. The following paragraph describes the universal composability (UC) framework on a very high level. We refer the reader to [Can01] for more details on the model.

In contrast to the stand-alone model for secure computation, the UC model allows the environment to be interactive. In particular, every message that is transmitted between parties is sent to the environment which can arbitrarily decide how to deal with it. The strongest possible adversary in this model is the so-called dummy adversary who simply forwards all instructions he receives from the environment. The UC framework offers a set of security-preserving composition theorems which allow for a modular analysis. If a protocol Π UC-realizes a functionality $\mathcal{F}$, a protocol Π' which uses protocol Π as a subroutine can be proven to securely realize a functionality $\mathcal{F}'$ in the $\mathcal{F}$-hybrid model, that is having access to the ideal functionality $\mathcal{F}$.

DEFINITION 18.2 (Adaptive UC-security, [Can01]). Let $\mathcal{A}$ be the dummy adversary and let θ denote the dummy protocol. We say a *protocol* Π *UC-realizes a functionality* $\mathcal{F}$ *in the* $\mathcal{G}$*-hybrid model in the presence of semi-honest adaptive adversaries* if for all PPT environments $\mathcal{Z}$, there exists a simulator Sim, such that

$$\left|\Pr\left[\mathsf{real}[\mathcal{Z},\mathcal{A},\pi^{\mathcal{G}}](\lambda)=1\right]-\Pr\left[\mathsf{ideal}[\mathcal{Z},\mathsf{Sim},\theta^{\mathcal{F}}](\lambda)=1\right]\right|$$

is negligible, where $\mathsf{real}[\mathcal{Z},\mathcal{A},\pi^{\mathcal{G}}](\lambda)$ denotes the output distribution of $\mathcal{Z}$ when interacting in the real world with $\mathcal{A}$ and $\pi^{\mathcal{G}}$ and $\mathsf{ideal}[\mathcal{Z},\mathsf{Sim},\theta^{\mathcal{F}}](\lambda)$ denotes the output distribution of $\mathcal{Z}$ when interacting in the ideal world with Sim and $\theta^{\mathcal{F}}$.

UC-security is a stronger notion than stand-alone security. More precisely, if there exists a protocol which UC-realizes a functionality according to

Definition 18.2, then there exists a protocol which securely computes that functionality according to Definition 18.1.

In the UC framework, the global CRS model corresponds to the $\mathcal{F}_{\mathsf{Setup}}^{\mathsf{crs}}$-hybrid model. The ideal functionality $\mathcal{F}_{\mathsf{Setup}}^{\mathsf{crs}}$ can be queried by any party and the adversary. Upon receiving such a query, $\mathcal{F}_{\mathsf{Setup}}^{\mathsf{crs}}$ gives the CRS *crs* to the querying party. If *crs* is not initialized, $\mathcal{F}_{\mathsf{Setup}}^{\mathsf{crs}}$ samples *crs* according to $\mathsf{Setup}(1^\lambda)$.

18.1.2 Pseudorandom Encodings Imply Adaptive MPC

AMPC for *deterministic* functionalities is possible assuming adaptively secure oblivious transfer due to [Kil88].

THEOREM 18.1 (informal, [Kil88; IPS08]). *Any deterministic functionality can be UC-realized in the OT-hybrid model in the presence of semi-honest adversaries adaptively corrupting any number of parties.*

Ishai et al. [IKOS10] prove that adaptively secure oblivious transfer in conjunction with $\mathsf{ISH}_{\approx_c}^{\mathsf{rand}}$ yields AMPC for all randomized functionalities in the plain model.

Assuming $\mathsf{acPREH}_{\approx_c}^{\mathsf{rand}}$, the strategy from [IKOS10] can be applied to show that every randomized functionality can be UC-realized in the (global CRS, OT)-hybrid model in the presence of semi-honest adversaries adaptively corrupting any number of parties.

THEOREM 18.2. *Assume* $\mathsf{acPREH}_{\approx_c}^{\mathsf{rand}}$ *holds, then any functionality* $\mathcal{F}$ *can be UC-realized in the (global CRS,* $\mathcal{F}_{\mathsf{OT}}$*)-hybrid-model, in the presence of semi-honest adaptive adversaries corrupting any number of parties.*

Proof. The proof follows the strategy from [IKOS10]. We consider functionalities $\mathcal{F}$ giving the same output to all parties. (This is without loss of generality since evaluating the function $\mathcal{F}'$ which distributes blinded individual outputs to the parties, i. e., $\mathcal{F}'((x_1,k_1),(x_2,k_2)) \coloneqq \mathcal{F}_1(x_1,x_2)\oplus k_1 \,\|\, \mathcal{F}_2(x_1,x_2)\oplus k_2$, yields the general case, see [GL91].) That is, the functionality $\mathcal{F}$ takes as input x_1, x_2 (the inputs of the parties P_1 and P_2, respectively) and internal randomness ρ, and outputs z to both parties. We consider the case of two parties (the proof easily extends to the case of many parties).

We view $\mathcal{F}$ as a PPT algorithm S. Due to $\mathsf{acPREH}_{\approx_c}^{\mathsf{rand}}$, there exist Setup_S, $\overline{S}$ and $\overline{S}^{-1}$ satisfying adaptive closeness and adaptive invertibility. Define a deterministic functionality $\mathcal{G}$ (in the $\mathcal{F}_{\text{g-crs}}^{\mathsf{Setup}_S}$-hybrid model) which takes inputs (x_1,ρ_1) from party P_1 and (x_2,ρ_2) from party P_2, and evaluates $\overline{S}(crs,(x_1,x_2);\rho_1\oplus\rho_2)$, where *crs* comes from $\mathcal{F}_{\text{g-crs}}^{\mathsf{Setup}_S}$. Due to Theorem 18.1, $\mathcal{G}$ can be UC-realized in the OT-hybrid model in the presence of semi-honest adversaries adaptively corrupting any number of parties.

Now we turn to realize $\mathcal{F}$ in the $(\mathcal{G},\mathcal{F}_{\text{g-crs}}^{\mathsf{Setup}_S})$-hybrid model. Party P_i chooses randomness ρ_i, feeds (x_i,ρ_i) into $\mathcal{G}$ and waits for the output. Without loss of generality, we assume that $\mathcal{Z}$ first observes the entire protocol before corrupting parties.[67] The simulator Sim works as follows. Sim initially only receives the output z of the ideal functionality $\mathcal{F}$ as input and calls $\mathcal{F}_{\text{g-crs}}^{\mathsf{Setup}_S}$ to obtain *crs*. If both parties are corrupted (first P_1, then P_2), Sim outputs a uniformly random string ρ_1 to explain the internal randomness of party P_1, computes $\rho \leftarrow \overline{S}^{-1}(crs,(x_1,x_2),z)$ and outputs $\rho_2 \coloneqq \rho\oplus\rho_1$ to explain the

[67] *This is because we consider semi-honest adversaries. Such adversaries collect as much information as possible without deviating from the protocol description. An adversary who corrupts parties earlier (or not at all) obtains strictly less information and hence simulation is easier.*

internal randomness of party P_2 (note that Sim learns the inputs x_1 and x_2 at the time of corrupting P_1 and P_2, respectively).

G_0

$crs \leftarrow \mathsf{Setup}_S(1^\lambda)$
$(x_1, x_2) \leftarrow \mathcal{Z}(crs)$
$\rho_1, \rho_2 \leftarrow \{0,1\}^{n(\lambda)}$
$z \leftarrow \mathcal{G}(x_1, \rho_1, x_2, \rho_2)$
upon corruption of P_1 **do**
  give (x_1, ρ_1, z) to $\mathcal{Z}$
upon corruption of P_2 **do**
  give (x_2, ρ_2) to $\mathcal{Z}$

(a) Description of G_0.

G_1

$crs \leftarrow \mathsf{Setup}_S(1^\lambda)$
$(x_1, x_2) \leftarrow \mathcal{Z}(crs)$
$\rho_1, \rho_2 \leftarrow \{0,1\}^{n(\lambda)}$
$z \leftarrow \overline{S}(crs, (x_1, x_2); \rho_1 \oplus \rho_2)$
upon corruption of P_1 **do**
  give (x_1, ρ_1, z) to $\mathcal{Z}$
upon corruption of P_2 **do**
  $\rho_2' := \rho_2$
  give (x_2, ρ_2') to $\mathcal{Z}$

(b) Description of G_1.

G_2

$crs \leftarrow \mathsf{Setup}_S(1^\lambda)$
$(x_1, x_2) \leftarrow \mathcal{Z}(crs)$
$\rho_1, \rho_2 \leftarrow \{0,1\}^{n(\lambda)}$
$z \leftarrow \overline{S}(crs, x_1, x_2; \rho_1 \oplus \rho_2)$
upon corruption of P_1 **do**
  give (x_1, ρ_1, z) to $\mathcal{Z}$
upon corruption of P_2 **do**
  $\rho_2' \leftarrow \rho_1 \oplus \overline{S}^{-1}(crs, (x_1, x_2), z)$
  give (x_2, ρ_2') to $\mathcal{Z}$

(c) Description of G_2.

G_3

$crs \leftarrow \mathsf{Setup}_S(1^\lambda)$
$(x_1, x_2) \leftarrow \mathcal{Z}(crs)$
$\rho_1 \leftarrow \{0,1\}^{n(\lambda)}$
$z \leftarrow S(x_1, x_2)$
upon corruption of P_1 **do**
  give (x_1, ρ_1, z) to $\mathcal{Z}$
upon corruption of P_2 **do**
  $\rho_2' \leftarrow \rho_1 \oplus \overline{S}^{-1}(crs, (x_1, x_2), z)$
  give (x_2, ρ_2') to $\mathcal{Z}$

(d) Description of G_3.

Figure 18.1: Hybrids used in proof of Theorem 18.2.

We proceed over a series of hybrids, see Figure 18.1.

GAME G_0. G_0 is identical to $\mathsf{real}[\mathcal{Z}, \mathcal{A}, \pi^{\mathcal{G}}](\lambda)$.

GAME G_1. G_1 is identical to G_0 except for the conceptional change that $\mathcal{G}$ is replaced with its implementation and ρ_2 is renamed to ρ_2' before being given to $\mathcal{Z}$. Hence, $|\Pr[out_0 = 1] - \Pr[out_1 = 1]|$.

GAME G_2. G_2 is identical to G_1 except that in G_1, $\mathcal{Z}$ receives the real random tape ρ_2 used by party P_2, whereas in G_2, $\mathcal{Z}$ receives ρ_2' which is computed as the logical XOR of ρ_1 and inverse sampled random coins $\overline{S}^{-1}(crs, (x_1, x_2), z)$.

CLAIM 18.1. For all PPT environments $\mathcal{Z}$, there exists a PPT adversary $\mathcal{B}_2$, such that $|\Pr[out_2 = 1] - \Pr[out_1 = 1]| \leqslant \mathrm{Adv}^{\text{a-crs-inv}}_{(\mathsf{Setup}_S, \overline{S}, \overline{S}^{-1}), \mathcal{B}_2}(\lambda)$.

Proof. Construct an adversary $\mathcal{B}_2$ breaking adaptive closeness as follows. $\mathcal{B}_2$ receives crs sampled from $\mathsf{Setup}_S(1^\lambda)$ and calls $\mathcal{Z}$ on input of crs. In

return, $\mathcal{B}_2$ receives (x_1, x_2) and samples $\rho_1 \leftarrow \{0,1\}^{n(\lambda)}$. Then, $\mathcal{B}_2$ outputs $m := (x_1, x_2)$ and receives (r, y) from the invertibility experiment. $\mathcal{B}_2$ uses $z := y$ and $\rho_2' := \rho_1 \oplus r$ and continues as in $\mathbf{G}_1$.

If (r, y) were produced such that r was sampled uniformly from $\{0,1\}^{n(\lambda)}$ and $y \leftarrow \overline{S}(crs, (x_1, x_2); r)$, then $\mathcal{B}_2$ perfectly simulates $\mathbf{G}_1$. If (r, y) were produced such that y was sampled from $\overline{S}(crs, (x_1, x_2))$ (using some uniform randomness) and r was inversely sampled as $r \leftarrow \overline{S}^{-1}(crs, (m_1, m_2), y)$, then $\mathcal{B}_2$ perfectly simulates $\mathbf{G}_2$. □

GAME $\mathbf{G}_3$. $\mathbf{G}_3$ is identical to $\mathbf{G}_2$ except that in $\mathbf{G}_2$, z is sampled using the alternative sampler $\overline{S}(crs, (x_1, x_2))$, whereas in $\mathbf{G}_3$, z is sampled using the original sampler $S(x_1, x_2)$. $\mathbf{G}_3$ is identical to $\mathsf{ideal}[\mathcal{Z}, \mathcal{A}, \pi^{\mathcal{G}}](\lambda)$.

CLAIM 18.2. For all PPT environments $\mathcal{Z}$, there exists a PPT adversary $\mathcal{B}_3$, such that $|\Pr[out_3 = 1] - \Pr[out_2 = 1]| \leqslant \mathrm{Adv}^{\text{a-crs-close}}_{(\mathsf{Setup}_S, \overline{S}, \overline{S}^{-1}), \mathcal{B}_3}(\lambda)$.

Proof. Construct an adversary $\mathcal{B}_3$ breaking adaptive closeness as follows. $\mathcal{B}_3$ receives crs sampled from $\mathsf{Setup}_S(1^\lambda)$ and calls $\mathcal{Z}$ on input of crs. In return, $\mathcal{B}_3$ receives (x_1, x_2), samples $\rho_1 \leftarrow \{0,1\}^{n(\lambda)}$ and outputs $m := (x_1, x_2)$ to the adaptive closeness experiment to receive y. $\mathcal{B}_3$ uses $z := y$ and continues as in $\mathbf{G}_2$. Note that $\mathbf{G}_2$ can be simulated without knowing the random coins used for $\overline{S}$.

If y was sampled using $\overline{S}(crs, (x_1, x_2))$, then $\mathcal{B}_3$ perfectly simulates $\mathbf{G}_2$. If y was sampled using $S(x_1, x_2)$, then $\mathcal{B}_3$ perfectly simulates $\mathbf{G}_3$. □

Thus, due to adaptive invertibility and adaptive closeness, for all environments $\mathcal{Z}$,

$$\left|\Pr\left[\mathsf{real}[\mathcal{Z}, \mathcal{A}, \pi^{\mathcal{G}}](\lambda) = 1\right] - \Pr\left[\mathsf{ideal}[\mathcal{Z}, \mathsf{Sim}, \theta^{\mathcal{F}}](\lambda) = 1\right]\right|$$

is negligible. □

Note that due to [DKR15] it is possible to realize UC-AMPC for every functionality only assuming the static version $\mathsf{cPREH}^{\mathsf{rand}}_{\approx_c}$ at the cost of additionally assuming the existence of an adaptively secure two-round oblivious transfer protocol. This can be avoided due to Corollary 16.1.

18.1.3 Adaptive MPC Implies Pseudorandom Encodings

In the following, we consider the standalone model [Canoo]. Note that in the standalone model, the environment is strictly weaker than in the UC-framework.

THEOREM 18.3. *If for all (two party) PPT functionalities $\mathcal{F}$, there exists a protocol Π that securely implements $\mathcal{F}$ in the global CRS model in the presence of adaptive semi-honest adversaries with post-execution corruption corrupting any number of parties, then* $\mathsf{acPREH}^{\mathsf{rand}}_{\approx_c}$ *(without universal setup) is true.*

Proof. The proof follows the ideas of [IKOS10]. Let S be an arbitrary PPT algorithm. Let x_1 denote the input of party P_1 and let x_2 denote the input of party P_2.

Consider the randomized functionality $\mathcal{F}$ which computes $z := S(x_1, x_2; \rho)$ and outputs z to P_1 and $\perp$ to P_2, where ρ is the internal random tape of the

functionality. We denote the message space of S by L. Let Π be a protocol which securely realizes $\mathcal{F}$ in the global CRS model (where the common reference string is drawn according some distribution $\mathsf{Setup}_{\mathcal{F}}$ possibly depending on the functionality $\mathcal{F}$) in the presence of adaptive semi-honest adversaries (with post-execution corruption). Further, let p_1 and p_2 be stateful PPT algorithms modeling the protocol messages exchanged between the parties P_1 and P_2. More formally, $(m_{2i+1}, z_i) \leftarrow p_1(crs, x_1, m_{2i}; r_1)$ and $m_{2j} \leftarrow p_2(crs, x_2, m_{2j-1}; r_2)$ for $i \geqslant 0, j \geqslant 1$ and $r_1, r_2 \leftarrow \{0,1\}^{p(\lambda)}$ (for some sufficiently large polynomial p). (Without loss of generality, we let p_1 additionally output z_i which equals $\perp$ during the protocol execution and contains the output of party P_1 after the execution.)

We define the invertible sampling scheme $(\mathsf{Setup}_S \coloneqq \mathsf{Setup}_{\mathcal{F}}, \overline{S}, \overline{S}^{-1})$ as in Figure 18.2.

$\overline{S}(crs, (x_1, x_2); (r_1, r_2))$

$z \coloneqq \perp, m \coloneqq \perp$
while $z = \perp$ **do**
  $(m, z) \leftarrow p_1(crs, x_1, m, r_1)$
  $m \leftarrow p_2(crs, x_2, m, r_2)$
return z

(a) The alternative sampler.

$\overline{S}^{-1}(crs, (x_1, x_2), z; r_S)$

$\{m'_i\}_{i \in [poly(\lambda)]} \leftarrow \mathsf{Sim}_1(crs; r_S)$
$(z, r'_1) \leftarrow \mathsf{Sim}_2(x_1, z)$
$r'_2 \leftarrow \mathsf{Sim}_3(x_2)$
return (r'_1, r'_2)

(b) The inverse sampler.

Figure 18.2: Definition of the alternative sampler $\overline{S}$ and the corresponding inverse sampler $\overline{S}^{-1}$.

LEMMA 18.1. *$(\mathsf{Setup}_S \coloneqq \mathsf{Setup}_{\mathcal{F}}, \overline{S}, \overline{S}^{-1})$ defined in Figure 18.2 satisfies adaptive closeness.*

Proof. Since we only consider protocols Π with at most polynomially many rounds, $\overline{S}$ is a PPT algorithm.

Intuitively, the output distributions of S and $\overline{S}$ are computationally close because of "correctness" of the protocol Π.

Let $\mathcal{B}$ be the adversary who only observes the protocol Π and does not interfere at all. By Definition 18.1, there exists a simulator Sim such that for all environments $\mathcal{Z}$,

$$\left| \Pr\left[crs \leftarrow \mathsf{Setup}_{\mathcal{F}}(1^\lambda) \colon \mathcal{Z}^{\mathcal{A},\Pi}(crs) = 1\right] - \Pr\left[crs \leftarrow \mathsf{Setup}_{\mathcal{F}}(1^\lambda) \colon \mathcal{Z}^{\mathsf{Sim},\mathcal{F}}(crs) = 1\right] \right|$$

is negligible. Note that since $\mathcal{B}$ corrupts no parties, Sim also corrupts no parties.

Let $\mathcal{A}$ be an adversary breaking closeness. We construct an environment $\mathcal{Z}$ which distinguishes between the real and the ideal world as follows. Initially, $\mathcal{Z}$ receives crs as input, calls $\mathcal{A}$ on input of crs and obtains $(x_1, x_2) \in L$. $\mathcal{Z}$ uses x_1 as input for party P_1 and x_2 as input for P_2. Then, $\mathcal{Z}$ executes the protocol (without any interference) and finally receives the output $(z, \perp)$. $\mathcal{Z}$ calls $\mathcal{A}$ on input z and outputs $\mathcal{A}$'s output.

In the ideal world, $\mathcal{Z}$ receives the outputs z and $\perp$ for P_1 and P_2 from the trusted third party T (since no party is corrupted) which are computed by $\mathcal{F}$.

Hence, if $\mathcal{Z}$ is in the ideal world, it simulates $\mathrm{Exp}^{(0)\text{-a-crs-close}}_{(\mathsf{Setup}_S,\overline{S},\overline{S}^{-1}),\mathcal{A}}(\lambda)$. In the real world, $\mathcal{Z}$ interacts with the actual parties running protocol Π and, hence, simulates $\mathrm{Exp}^{(1)\text{-a-crs-close}}_{(\mathsf{Setup}_S,\overline{S},\overline{S}^{-1}),\mathcal{A}}(\lambda)$. Therefore,

$$\mathrm{Adv}^{\text{a-crs-close}}_{(\mathsf{Setup}_S,\overline{S},\overline{S}^{-1}),\mathcal{A}}(\lambda) \leqslant \Big|\ \Pr\left[crs \leftarrow \mathsf{Setup}_{\mathcal{F}}(1^\lambda)\colon \mathcal{Z}^{\mathcal{A},\Pi}(crs)=1\right] - \Pr\left[crs \leftarrow \mathsf{Setup}_{\mathcal{F}}(1^\lambda)\colon \mathcal{Z}^{\mathsf{Sim},\mathcal{F}}(crs)=1\right]\Big|$$

for the environment $\mathcal{Z}$ and the adversary $\mathcal{B}$ which do not interfere with the protocol execution. □

Let $\mathcal{B}$ be an adversary that operates as follows. During the protocol, $\mathcal{B}$ records the transcripts $\{m_i\}_{i\in[poly'(\lambda)]}$ and outputs them. At the end of the protocol, $\mathcal{B}$ corrupts P_1 obtaining the input x_1, output z and random tape r_1. $\mathcal{B}$ directly outputs (z, r_1). Afterwards, $\mathcal{B}$ (post-execution) corrupts P_2 and obtains the input x_2 and the random tape r_2 and outputs r_2.

Since we assume Π securely realizes $\mathcal{F}$ (in particular in the presence of $\mathcal{B}$), there exists a (stateful) simulator $\mathsf{Sim} := (\mathsf{Sim}_1, \mathsf{Sim}_2, \mathsf{Sim}_3)$ producing outputs that are computationally indistinguishable from the outputs $\mathcal{B}$ produces. Sim_1 simulates the execution before a corruption occurs. On input of *crs* and a random tape r_S, Sim_1 produces $\{m'_i\}_{i\in[poly'(\lambda)]}$. On corruption of P_1 (in the post-execution corruption stage), Sim_2 obtains x_1, z and outputs a string r'_1 corresponding to the randomness that P_1 used to produce its messages in the protocol (using input x_1) and outputs (z, r'_1). On corruption of P_2 (in the post-execution corruption stage), Sim_3 obtains x_2 and outputs a string r'_2 corresponding to the randomness that P_2 used to produce its messages (using input x_2).

LEMMA 18.2. $(\mathsf{Setup}_S := \mathsf{Setup}_{\mathcal{F}}, \overline{S}, \overline{S}^{-1})$ *defined in Figure 18.2 satisfies adaptive invertibility.*

Proof. Since Sim_1, Sim_2 and Sim_3 are PPT algorithms, $\overline{S}^{-1}$ is a PPT algorithm.

Recall that $\mathcal{F}(x_1, x_2; \rho) := S(x_1, x_2; \rho)$. Define a modified functionality $\mathcal{F}'(x_1, x_2; \rho') := \overline{S}(x_1, x_2; \rho')$. We consider an intermediate game, where $\mathcal{Z}$ interacts with the simulator Sim and the modified ideal functionality $\mathcal{F}'$.

CLAIM 18.3. For all PPT environments $\mathcal{Z}$,

$$\Big|\ \Pr\left[crs \leftarrow \mathsf{Setup}_{\mathcal{F}}(1^\lambda)\colon \mathcal{Z}^{\mathsf{Sim},\mathcal{F}}(crs)=1\right] - \Pr\left[crs \leftarrow \mathsf{Setup}_{\mathcal{F}}(1^\lambda)\colon \mathcal{Z}^{\mathsf{Sim},\mathcal{F}'}(crs)=1\right]\Big|$$

is negligible.

Proof of Claim 18.3. Let $\mathcal{Z}$ be a PPT environment. We construct an adversary $\mathcal{B}$ breaking adaptive closeness. $\mathcal{B}$ simulates the entire interaction between $\mathcal{Z}$, the simulator Sim and the trusted party T, which evaluates the functionality $\mathcal{F}$ (or $\mathcal{F}'$). In particular, $\mathcal{B}$ takes control of the party T. On input of *crs*, $\mathcal{B}$ simulates the interaction between $\mathcal{Z}$ and Sim (given *crs*) until the computation stage. In the computation stage, the parties P_1 and P_2 send their inputs (which may have been adaptively chosen based on *crs*) to $\mathcal{B}$. $\mathcal{B}$ outputs (x_1, x_2) to $\mathrm{Exp}^{(b)\text{-a-crs-close}}_{(\mathsf{Setup}_S,\overline{S},\overline{S}^{-1}),\mathcal{A}}(\lambda)$ and receives an output $z := y_b$, where $y_0 \leftarrow S(x_1, x_2)$ and $y_1 \leftarrow \overline{S}(x_1, x_2)$. $\mathcal{B}$ sends z to P_1 and P_2 and continues to simulate the

interaction between $\mathcal{Z}$ and Sim. This is possible, because the randomness which is actually used by T is necessary for simulation. Finally, $\mathcal{Z}$ halts and outputs a bit b'. $\mathcal{B}$ outputs b'. □

Let $\mathcal{A}$ be an adversary breaking adaptive invertibility. We construct an environment $\mathcal{Z}$ which distinguishes between the real execution of Π with adversary $\mathcal{B}$ and the ideal execution with Sim and the ideal functionality $\mathcal{F}'$. Initially, $\mathcal{Z}$ receives *crs* and calls $\mathcal{A}$ on input of *crs* to obtain (x_1, x_2). $\mathcal{Z}$ uses x_1 as input for P_1 and x_2 as input for P_2. After the execution of the protocol and the post-execution corruptions, $\mathcal{Z}$ receives $(r_1, r_2), z$ (either by the adversary $\mathcal{B}$ or the simulator Sim). Finally, $\mathcal{Z}$ calls $\mathcal{A}$ on input of $((r_1, r_2), z)$ and outputs $\mathcal{A}$'s output.

In the real world, z is the output of the real protocol and r_1, r_2 is the actual randomness used by the parties. Hence, by definition of $\overline{S}$, (r_1, r_2) is the randomness actually used by $\overline{S}$ to produce the output z. Therefore,

$$\Pr\left[\mathrm{Exp}^{(0)\text{-a-crs-inv}}_{(\mathsf{Setup}_S, \overline{S}, \overline{S}^{-1}), \mathcal{A}}(\lambda) = 1\right] = \Pr_{crs}\left[\mathcal{Z}^{\mathcal{B}, \Pi}(crs) = 1\right].$$

In the ideal world, z is produced by the functionality $\mathcal{F}'$ (hence, by $\overline{S}$) and $(r_1, r_2) := (r_1', r_2')$ is produced by $\overline{S}^{-1}((x_1, x_2), z)$. Hence,

$$\Pr\left[\mathrm{Exp}^{(1)\text{-a-crs-inv}}_{(\mathsf{Setup}_S, \overline{S}, \overline{S}^{-1}), \mathcal{A}}(\lambda) = 1\right] = \Pr_{crs}\left[\mathcal{Z}^{\mathsf{Sim}, \mathcal{F}'}(crs) = 1\right].$$

Hence,

$$\begin{aligned}
&\mathrm{Adv}^{\text{a-crs-inv}}_{(\mathsf{Setup}_S, \overline{S}, \overline{S}^{-1}), \mathcal{A}}(\lambda) \\
&\quad = \left|\Pr_{crs}\left[\mathcal{Z}^{\mathcal{B}, \Pi}(crs) = 1\right] - \Pr_{crs}\left[\mathcal{Z}^{\mathsf{Sim}, \mathcal{F}'}(crs) = 1\right]\right| \\
&\quad \leqslant \left|\Pr_{crs}\left[\mathcal{Z}^{\mathcal{B}, \Pi}(crs) = 1\right] - \Pr_{crs}\left[\mathcal{Z}^{\mathsf{Sim}, \mathcal{F}}(crs) = 1\right]\right| \\
&\qquad + \left|\Pr_{crs}\left[\mathcal{Z}^{\mathsf{Sim}, \mathcal{F}}(crs) = 1\right] - \Pr_{crs}\left[\mathcal{Z}^{\mathsf{Sim}, \mathcal{F}'}(crs) = 1\right]\right|
\end{aligned}$$

which is negligible by assumption and Claim 18.3. □

Therefore, for every PPT algorithm S, there exists an algorithm Setup_S, an alternative sampler $\overline{S}$ and an inverse sampler $\overline{S}^{-1}$ such that adaptive closeness and adaptive invertibility hold. This concludes the proof. □

Combining Theorems 18.2 and 18.3 and since semi-honest adaptive MPC in the non-programmable common random string model implies semi-honest adaptive MPC in the plain model, we obtain the following corollary improving Corollary 17.9.

COROLLARY 18.1. *If there exist extractable one-way function family ensembles (without auxiliary information) and indistinguishability obfuscation for all circuits, then* weak $\mathsf{cPREH}^{\mathsf{rand}}_{\approx_c}$ *with common random string (URC) is false.*

18.2 HONEY ENCRYPTION

Honey encryption schemes [JR14] are secret-key encryption schemes which offer security even against unbounded adversaries, i. e., even against adversaries who are able to perform an exhaustive search over all secret-keys.

This is a particularly useful notion when the key comes from a distribution with low min-entropy, like passwords. On decryption with a wrong key, a honey encryption scheme behaves indistinguishably from decryption with the actually used key.

Let D_1 be an efficiently samplable key distribution over the key space $\mathcal{K}_\lambda$ and let D_2 be an efficiently samplable message distribution over the message space $\mathcal{M}_\lambda$.

DEFINITION 18.3 (Honey encryption, [JR14]). A secret-key encryption (SKE) scheme $(\mathsf{Enc}, \mathsf{Dec})$ is called *honey encryption (HE) scheme for key distribution* D_1 *and plaintext distribution* D_2 if it satisfies the following property.

SECURITY AGAINST MESSAGE RECOVERY. The encryption scheme $(\mathsf{Enc}, \mathsf{Dec})$ is secure against message recovery (with respect to D_1, D_2) if for all unbounded adversaries $\mathcal{A}$, the advantage

$$\mathrm{Adv}^{\mathsf{mr}}_{(\mathsf{Enc},\mathsf{Dec}),\mathcal{A},D_1,D_2}(\lambda) = \Pr\left[\mathrm{Exp}^{\mathsf{mr}}_{(\mathsf{Enc},\mathsf{Dec}),\mathcal{A},D_1,D_2}(\lambda) = 1\right]$$

is negligibly close to $2^{-\mu_1}$, where μ_1 is the min-entropy of D_1 (i. e., $\mu_1 = H_\infty(D_1)$), and $\mathrm{Exp}^{\mathsf{mr}}_{(\mathsf{Enc},\mathsf{Dec}),\mathcal{A},D_1,D_2}(\lambda)$ is defined as in Figure 18.3.

$\mathrm{Exp}^{\mathsf{mr}}_{(\mathsf{Enc},\mathsf{Dec}),\mathcal{A},D_1,D_2}(\lambda)$

$k \leftarrow D_1(1^\lambda)$

$m^* \leftarrow D_2(1^\lambda)$

$c^* \leftarrow \mathsf{Enc}(k, m^*)$

$m \leftarrow \mathcal{A}(c^*)$

return $m = m^*$

Figure 18.3: Message recovery game.

DEFINITION 18.4 (Distribution-transforming encoder, [JR14; JRT16]). A *distribution-transforming* encoder (DTE) for a distribution sampled by a sampler S over $\mathcal{Y}_{S,\lambda}$ is a tuple of efficient algorithms (E_S, D_S), where D_S is deterministic, such that the following properties are satisfied.

PERFECT CORRECTNESS. For all $y \in \mathcal{Y}_{S,\lambda}$, $\Pr\left[D_S(E_S(m)) = m\right] = 1$, where the probability is over the randomness of E_S.

DTE SECURITY. For all unbounded adversaries,

$$\mathrm{Adv}^{\mathsf{dte}}_{(E_S,D_S),\mathcal{A}}(\lambda) \coloneqq \left| \Pr\left[\mathrm{Exp}^{(0)\text{-}\mathsf{dte}}_{(E_S,D_S),\mathcal{A}}(\lambda) = 1\right] - \Pr\left[\mathrm{Exp}^{(1)\text{-}\mathsf{dte}}_{(E_S,D_S),\mathcal{A}}(\lambda) = 1\right] \right|$$

is negligible, where $\mathrm{Exp}^{(0)\text{-}\mathsf{dte}}_{(E_S,D_S),\mathcal{A}}(\lambda)$ and $\mathrm{Exp}^{(1)\text{-}\mathsf{dte}}_{(E_S,D_S),\mathcal{A}}(\lambda)$ are defined in Figure 18.4.

The above notions are defined with respect to unbounded adversaries. This is a simplification meant to capture the fact that an adversary is able to perform work proportional to $2^{-\mu_1}$. For low-entropic key distributions, a computational variant of the above definition suffices. We refer to this as *computational* honey encryption.

(a) DTE game using an encoding of a sample.

(b) DTE game using a uniform random string.

Figure 18.4: DTE games.

Relaxing the perfect correctness requirement from Definition 18.4 recovers the definition of (weak) $\mathsf{PREH}^{\mathsf{rand}}_{\equiv_s}$ (or weak $\mathsf{PREH}^{\mathsf{rand}}_{\approx_c}$ in the computational case) by Lemma 17.10 and still suffices to imply honey encryption via the DTE-then-Encrypt framework due to [JR14] given a secret-key encryption scheme (with message space matching the range of E_S). Applying the DTE-then-Encrypt framework on $\mathsf{acPREH}^{\mathsf{rand}}_{\equiv_s}$ and $\mathsf{acPREH}^{\mathsf{rand}}_{\approx_c}$ yields honey encryption in a CRS model for high-entropic and low-entropic key distributions, respectively. Note that the adaptive versions are only necessary when considering message distributions D_2 with input.

THEOREM 18.4. *If* $\mathsf{acPREH}^{\mathsf{rand}}_{\equiv_s}$ *(or* $\mathsf{acPREH}^{\mathsf{rand}}_{\approx_c}$*) is true and a suitable secret-key encryption scheme (modeled as random cipher) exists, then (computational) honey encryption exists for all efficiently samplable message distributions in the CRS model.*

The proof directly follows from [JR14]. Together with Theorems 17.13 and 17.14, we obtain the following corollary.

COROLLARY 18.2. *If polynomially secure IO exists and a (suitable) secret-key encryption scheme (modeled as random cipher) exists, then computational honey encryption exists for all efficiently samplable message distributions in the CRS model.*

18.3 KEYLESS STEGANOGRAPHY

Pseudorandom encodings yield a notion of keyless steganography. In the case of $\mathsf{acPREH}^{\mathsf{rand}}_{\approx_c}$ all parties need access to some public parameters, but none of the parties needs access to a secret key. We adopt the notation from [BL18b]. Since decoding does not involve any secret information, any attack corresponds to an equivalent of SS-CCA-security for public-key stegosystems (PKStS), where the decoding may even be applied on the challenge. Hence, the definition of SS-CCA-security must be adapted such that the message to be hidden is not chosen by the adversary but sampled according to a predefined message distribution. Let $dl(\lambda)$ be the document length and $ol(\lambda)$ be the output length. Let $\mathcal{C}_{\lambda,dl(\lambda)}$ be a channel which defines a probability distribution over $\Sigma^{dl(\lambda)}$ depending on the history $(\Sigma^{dl(\lambda)})^*$.

DEFINITION 18.5 (Keyless stegosystem for distribution S). *A keyless stegosystem (KlStS) for message distribution* S is a triple of PPT algorithms $(\mathsf{KlStS.Setup}, \mathsf{KlStS.E}_S, \mathsf{KlStS.D}_S)$, where

- $\mathsf{KlStS.Setup}(1^\lambda)$ produces public parameters pp (without corresponding secret information),

$\mathrm{Exp}^{(b)\text{-klsts-sec}}_{\mathsf{KlStS},\mathcal{A},\mathcal{C}}(\lambda)$

$pp \leftarrow \mathsf{KlStS.Setup}(1^\lambda)$
$m \leftarrow \mathcal{A}(pp)$
$m^* \leftarrow S(m)$
$hist^* \leftarrow \mathcal{A}(1^\lambda, pp, m^*)$
$\mathbf{d}^*_0 \leftarrow \mathsf{KlStS.E}^{\mathcal{C}}_S(pp, m^*, hist^*)$
$\mathbf{d}^*_1 \leftarrow \mathcal{C}^{ol(\lambda)}_{\lambda, dl(\lambda), hist^*}$
return $\mathcal{A}(\mathbf{d}^*_b)$

Figure 18.5: Definition of the security game for keyless stegosystems.

- $\mathsf{KlStS.E}_S$ on input of pp, a message y sampled from S, a history $hist \in (\Sigma^{dl(\lambda)})^*$ and some state information $s \in \{0,1\}^*$, produces a document $d \in \Sigma^{dl}$ (by being able to sample from $C_{\lambda, dl(\lambda)}$). $\mathsf{KlStS.E}^{\mathcal{C}}_S(pp, m, hist)$ denotes sampling $ol(\lambda)$ documents using $\mathsf{KlStS.E}_S$ one-by-one.
- $\mathsf{KlStS.D}_S$ on input of pp, and a sequence of documents $d_1, \ldots, d_{ol(\lambda)}$, outputs a message m'.

We require KlStS to meet the following properties.

UNIVERSALITY. KlStS works on any channel without prior knowledge of the distribution of the channel.

RELIABILITY. The probability

$$\Pr\left[\mathsf{KlStS.D}_S\left(pp, \mathsf{KlStS.E}_S(pp, m, hist), hist\right) \neq m\right]$$

is negligible, where the probability is over the choice of pp, m and the random coins of $\mathsf{KlStS.E}_S$.

SECURITY. KlStS is secure (on channel $\mathcal{C}$), if for all PPT adversaries $\mathcal{A}$,

$$\mathrm{Adv}^{\text{klsts-sec}}_{\mathsf{KlStS},\mathcal{A},\mathcal{C}}(\lambda) := \left| \Pr\left[\mathrm{Exp}^{(0)\text{-klsts-sec}}_{\mathsf{KlStS},\mathcal{A},\mathcal{C}}(\lambda) = 1\right] - \Pr\left[\mathrm{Exp}^{(1)\text{-klsts-sec}}_{\mathsf{KlStS},\mathcal{A},\mathcal{C}}(\lambda) = 1\right] \right|$$

is negligible, where $\mathrm{Exp}^{(0)\text{-klsts-sec}}_{\mathsf{KlStS},\mathcal{A},\mathcal{C}}(\lambda)$ and $\mathrm{Exp}^{(1)\text{-klsts-sec}}_{\mathsf{KlStS},\mathcal{A},\mathcal{C}}(\lambda)$ are defined in Figure 18.5.

Applying a similar strategy as in [vH04; Hop05], we obtain the following theorem.

THEOREM 18.5. *Let S be an efficiently samplable message distribution and let $\mathcal{H}$ be a family of pairwise independent hash functions. If $\mathsf{acPREH}^{\mathsf{rand}}_{\approx_c}$ is true for S, then $(\mathsf{KlStS.Setup}_S, \mathsf{KlStS.E}_S, \mathsf{KlStS.D}_S)$ defined in Figure 18.6 is a keyless stegosystem for the message distribution S.*

The result basically follows from the Leftover Hash Lemma [HILL99] and the ability to embed the message distribution into the uniform distribution due to $\mathsf{acPREH}^{\mathsf{rand}}_{\approx_c}$. Note that since in $\mathrm{Exp}^{(b)\text{-klsts-sec}}_{\mathsf{KlStS},\mathcal{A},\mathcal{C}}(\lambda)$, the adversary does not know the challenge message m^*, it is not necessary that each encoding of a message corresponds to exactly one stegotext, see [Hop05]. The system is universal since it does not assume any knowledge on the channel.

$\mathsf{KlStS.Setup}_S(1^\lambda)$

$crs \leftarrow \mathsf{Setup}_S(1^\lambda)$

$H \leftarrow \mathcal{H}$

return $pp := (crs, H)$

(a) Setup algorithm.

$\mathsf{KlStS.E}_S((crs, H), m)$

$u \leftarrow \mathsf{E}_S(crs, m)$

$u_1 \| \ldots \| u_\ell =: u$

for $i \in [\ell]$ **do**

  do

    $d_i \leftarrow \mathcal{C}(\lambda, hist)$

  until $prefix_m(H(d_i)) = u_i$

return $(d_1, \ldots, d_\ell)$

(b) Encode algorithm.

$\mathsf{KlStS.D}_S((crs, H), (d_1, \ldots, d_\ell))$

for $i \in [\ell]$ **do**

  $u_i \leftarrow prefix_m(H(d_i))$

$u \leftarrow u_1 \| \ldots \| u_\ell$

return $\mathsf{D}_S(crs, u)$

(c) Decode algorithm.

Figure 18.6: Description of a keyless stegosystem from pseudorandom encodings inspired by [vH04; Hop05].

18.4 DENIABLE ENCRYPTION

As noted in [CDNO97], sender deniable encryption is related to adaptively secure MPC (for adaptively well-formed functionalities). In this section, we analyze if $\mathsf{cPREH}^{\mathsf{rand}}_{\approx_c}$ together with the existence of a PKE scheme suffices to obtain deniable encryption.

Recall, that the explainability compiler of [DKR15] which is based on the deniable encryption scheme of [SW14] corresponds to $\mathsf{cISH}^{\mathsf{rand}}_{\approx_c}$, where closeness actually holds information-theoretically. Let cISH' denote this variant of $\mathsf{cISH}^{\mathsf{rand}}_{\approx_c}$, where closeness holds information-theoretically.

DEFINITION 18.6 (Publicly deniable encryption, [SW14]). A publicly deniable encryption (DE) scheme for message space $\mathcal{M}$ is a tuple $\mathsf{DE} = (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Explain})$ such that $(\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ is an IND-CPA secure PKE scheme and the following property is satisfied.

INDISTINGUISHABILITY OF EXPLANATION. For all PPT adversaries $\mathcal{A}$,

$$\mathrm{Adv}^{\mathsf{ind\text{-}expl}}_{\mathsf{DE},\mathcal{A}}(\lambda) := \left| \Pr\left[\mathrm{Exp}^{(0)\text{-}\mathsf{ind\text{-}expl}}_{\mathsf{DE},\mathcal{A}}(\lambda) = 1\right] - \Pr\left[\mathrm{Exp}^{(1)\text{-}\mathsf{ind\text{-}expl}}_{\mathsf{DE},\mathcal{A}}(\lambda) = 1\right] \right|$$

is negligible, where $\mathrm{Exp}^{(0)\text{-}\mathsf{ind\text{-}expl}}_{\mathsf{DE},\mathcal{A}}(\lambda)$ and $\mathrm{Exp}^{(1)\text{-}\mathsf{ind\text{-}expl}}_{\mathsf{DE},\mathcal{A}}(\lambda)$ are defined in Figure 18.7 and the randomness space of Enc is $\{0,1\}^{n(\lambda)}$.

We only consider publicly deniable encryption schemes with message space $\{0,1\}$, since a deniable encryption scheme with message space $\{0,1\}$ implies a publicly deniable encryption scheme for message space $\{0,1\}^n$ (for a polynomial n in λ).

$\mathrm{Exp}^{(0)\text{-ind-expl}}_{\mathsf{DE},\mathcal{A}}(\lambda)$

$(pk, sk) \leftarrow \mathsf{KGen}(1^\lambda)$
$m^* \leftarrow \mathcal{A}(pk)$
$u^* \leftarrow \{0,1\}^{n(\lambda)}$
$c^* \leftarrow \mathsf{Enc}(pk, m^*; u^*)$
return $\mathcal{A}(c^*, u^*)$

(a) Adaptive indistinguishability of explanation game using true randomness.

$\mathrm{Exp}^{(1)\text{-ind-expl}}_{\mathsf{DE},\mathcal{A}}(\lambda)$

$(pk, sk) \leftarrow \mathsf{KGen}(1^\lambda)$
$m^* \leftarrow \mathcal{A}(pk)$
$u^* \leftarrow \{0,1\}^{n(\lambda)}$
$c^* \leftarrow \mathsf{Enc}(pk, m^*; u^*)$
$e^* \leftarrow \mathsf{Explain}(pk, c^*)$
return $\mathcal{A}(c^*, e^*)$

(b) Adaptive indistinguishability of explanation game using explained randomness.

$\mathrm{Exp}^{(0)\text{-ind-expl}'}_{\mathsf{DE},\mathcal{A}}(\lambda)$

$m^* \leftarrow \mathcal{A}(1^\lambda)$
$(pk, sk) \leftarrow \mathsf{KGen}(1^\lambda)$
$u^* \leftarrow \{0,1\}^{n(\lambda)}$
$c^* \leftarrow \mathsf{Enc}(pk, m^*; u^*)$
return $\mathcal{A}(pk, c^*, u^*)$

(c) Static indistinguishability of explanation game using true randomness.

$\mathrm{Exp}^{(1)\text{-ind-expl}'}_{\mathsf{DE},\mathcal{A}}(\lambda)$

$m^* \leftarrow \mathcal{A}(1^\lambda)$
$(pk, sk) \leftarrow \mathsf{KGen}(1^\lambda)$
$u^* \leftarrow \{0,1\}^{n(\lambda)}$
$c^* \leftarrow \mathsf{Enc}(pk, m^*; u^*)$
$e^* \leftarrow \mathsf{Explain}(pk, c^*)$
return $\mathcal{A}(pk, c^*, e^*)$

(d) Static indistinguishability of explanation game using explained randomness.

Figure 18.7: Indistinguishability of explanation games in its adaptive (Figures 18.7a and 18.7b) and static (Figures 18.7c and 18.7d) variant.

For message space $\{0,1\}$, indistinguishability of explanation is equivalent to static indistinguishability of explanation, i. e., indistinguishability between $\mathrm{Exp}^{(0)\text{-ind-expl}'}_{\mathsf{DE},\mathcal{A}}(\lambda)$ and $\mathrm{Exp}^{(1)\text{-ind-expl}'}_{\mathsf{DE},\mathcal{A}}(\lambda)$ as defined in Figure 18.7.

cISH' in conjunction with a PKE scheme yields (publicly) sender deniable encryption.

THEOREM 18.6. *Let* $(\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ *be an IND-CPA-secure public-key encryption scheme for message space* $\{0,1\}$. *If* cISH' *holds, then there exists a publicly deniable encryption scheme for message space* $\{0,1\}$.

Proof. cISH' implies that for the PPT algorithm Enc there exist a setup algorithm $\mathsf{Setup}_{\mathsf{Enc}}$, an alternative sampler $\overline{\mathsf{Enc}}$ and an inverse sampler $\overline{\mathsf{Enc}}^{-1}$ satisfying statistical closeness and computational invertibility. We define a publicly deniable encryption scheme $(\mathsf{KGen}', \mathsf{Enc}', \mathsf{Dec}', \mathsf{Explain}')$ in Figure 18.8.

CORRECTNESS. Let $m \in \{0,1\}$ be a plaintext.

$$\epsilon_1 := \Pr\left[\begin{matrix}((crs, pk), sk) = (pk', sk') \leftarrow \mathsf{KGen}'(1^\lambda) \\ c \leftarrow \mathsf{Enc}(pk, m)\end{matrix} : \mathsf{Dec}(sk, c) \neq m\right]$$

$$\epsilon_2 := \Pr\left[\begin{matrix}((crs, pk), sk) = (pk', sk') \leftarrow \mathsf{KGen}'(1^\lambda) \\ c \leftarrow \overline{\mathsf{Enc}}(pk, m)\end{matrix} : \mathsf{Dec}(sk, c) \neq m\right]$$

$\mathsf{KGen}'(1^\lambda)$

$crs \leftarrow \mathsf{Setup}_{\mathsf{Enc}}(1^\lambda)$
$(pk, sk) \leftarrow \mathsf{KGen}(1^\lambda)$
$pk' \coloneqq (crs, pk), sk' \coloneqq sk$
return (pk', sk')

(a) The key generation algorithm.

$\mathsf{Enc}'(pk', \mathrm{m})$

parse $(crs, pk) =: pk'$
$\mathrm{c} \leftarrow \overline{\mathsf{Enc}}(crs, (pk, \mathrm{m}))$
return c

(b) The encryption algorithm.

$\mathsf{Dec}'(sk, \mathrm{c})$

$\mathrm{m} \leftarrow \mathsf{Dec}(sk, \mathrm{c})$
return m

(c) The decryption algorithm.

$\mathsf{Explain}'(pk', \mathrm{m})$

parse $(crs, pk) =: pk'$
$\mathrm{u} \leftarrow \overline{\mathsf{Enc}}^{-1}(crs, (pk, \mathrm{m}), \mathrm{c})$
return u

(d) The explain algorithm.

Figure 18.8: Publicly deniable encryption scheme from cISH′.

Consider an *unbounded* adversary $\mathcal{A}$ on closeness which on input of (crs, pk, c), computes sk (using exhaustive search) and outputs 1 if $\mathsf{Dec}(sk, \mathrm{c}) \neq \mathrm{m}$ and 0 otherwise. The advantage of this adversary is

$$\mathrm{Adv}^{\text{crs-close}}_{(\mathsf{Setup}_{\mathsf{Enc}}, \overline{\mathsf{Enc}}, \overline{\mathsf{Enc}}^{-1}), \mathcal{A}}(\lambda) = |\epsilon_1 - \epsilon_2|.$$

Hence,

$$\epsilon_2 \leqslant \mathrm{Adv}^{\text{crs-close}}_{(\mathsf{Setup}_{\mathsf{Enc}}, \overline{\mathsf{Enc}}, \overline{\mathsf{Enc}}^{-1}), \mathcal{A}}(\lambda) + \epsilon_1$$

and therefore negligible due to statistical closeness and correctness of $(\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$.

IND-CPA SECURITY. For message space $\{0,1\}$, IND-CPA security of $(\mathsf{KGen}', \mathsf{Enc}', \mathsf{Dec}')$ is equivalent to the indistinguishability between the games $\mathbf{G}_0$ and $\mathbf{G}_3$ of Figure 18.9.

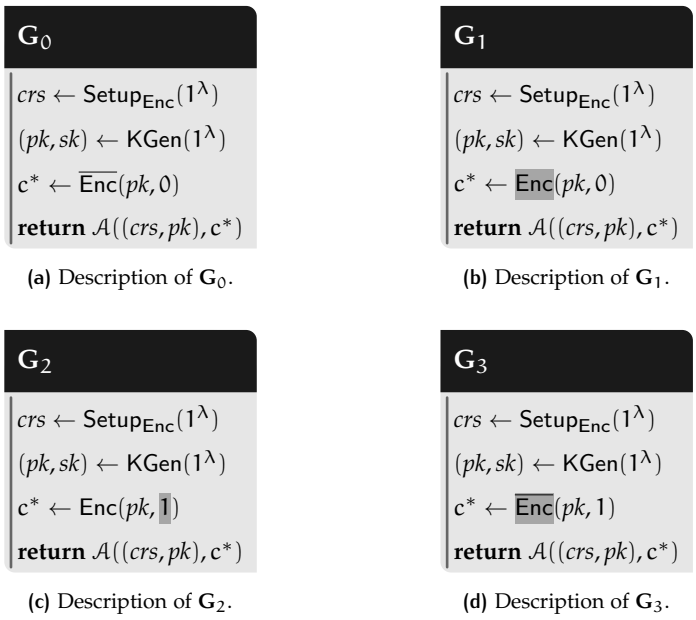
GAME $\mathbf{G}_0$. $\mathbf{G}_0$ generates a key pair as in $\mathsf{DE.KGen}(1^\lambda)$, encrypts 0 using DE.Enc and gives the public key and the ciphertext to the adversary $\mathcal{A}$.

GAME $\mathbf{G}_1$. $\mathbf{G}_1$ is identical to $\mathbf{G}_0$ except that c^* is produced using the original sampler Enc instead of the alternative sampler $\overline{\mathsf{Enc}}$.

CLAIM 18.4. For all (unbounded) adversaries $\mathcal{A}$ there exists an (unbounded) adversary $\mathcal{B}_1$ such that

$$\left|\Pr\left[out_1 = 1\right] - \Pr\left[out_0 = 1\right]\right| \leqslant \mathrm{Adv}^{\text{crs-close}}_{(\mathsf{Setup}_{\mathsf{Enc}}, \overline{\mathsf{Enc}}, \overline{\mathsf{Enc}}^{-1}), \mathcal{B}_1}(\lambda).$$

Proof. Let $\mathcal{A}$ be an unbounded adversary distinguishing $\mathbf{G}_0$ and $\mathbf{G}_1$. Construct an adversary $\mathcal{B}_1$ breaking closeness. Initially, $\mathcal{B}_1$ samples $(pk, sk) \leftarrow \mathsf{KGen}(1^\lambda)$ and outputs $\mathrm{m} \coloneqq (pk, 0)$ to the experiment. In return, $\mathcal{B}_1$ receives



(a) Description of $\mathbf{G}_0$. **(b)** Description of $\mathbf{G}_1$.

(c) Description of $\mathbf{G}_2$. **(d)** Description of $\mathbf{G}_3$.

Figure 18.9: Hybrids used in the proof of IND-CPA security for Theorem 18.6.

(crs, y), where y is either sampled using $\mathsf{Enc}(m)$ or $\overline{\mathsf{Enc}}(m)$. $\mathcal{B}_1$ calls $\mathcal{A}$ on input of $((crs, pk), y)$. Hence,

$$\Pr\left[out_0 = 1\right] = \Pr\left[\mathrm{Exp}^{(1)\text{-crs-close}}_{(\mathsf{Setup}_{\mathsf{Enc}}, \overline{\mathsf{Enc}}, \overline{\mathsf{Enc}}^{-1}), \mathcal{B}_1}(\lambda) = 1\right],$$

$$\Pr\left[out_1 = 1\right] = \Pr\left[\mathrm{Exp}^{(0)\text{-crs-close}}_{(\mathsf{Setup}_{\mathsf{Enc}}, \overline{\mathsf{Enc}}, \overline{\mathsf{Enc}}^{-1}), \mathcal{B}_1}(\lambda) = 1\right].$$

□

GAME $\mathbf{G}_2$. $\mathbf{G}_2$ is identical to $\mathbf{G}_1$ except that Enc is used to encrypt 1 instead of 0. This game hop is justified by the IND-CPA security of $(\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$.

GAME $\mathbf{G}_3$. $\mathbf{G}_3$ is identical to $\mathbf{G}_2$ except that c^* is produced using the alternative sampler $\overline{\mathsf{Enc}}$ instead of the original sampler Enc. This game hop is justified by statistical closeness like the game hop from $\mathbf{G}_0$ to $\mathbf{G}_1$.

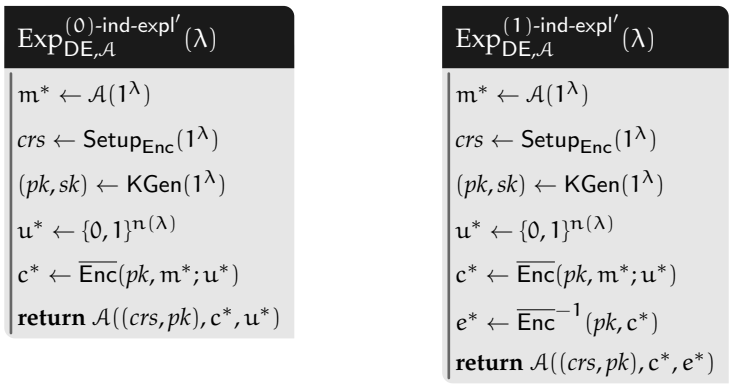
INDISTINGUISHABILITY OF EXPLANATION. We prove indistinguishability between the games described in Figure 18.10.

CLAIM 18.5. For all PPT adversaries $\mathcal{A}$, $\mathrm{Adv}^{\text{ind-expl}'}_{\mathsf{DE},\mathcal{A}}(\lambda)$ is negligible.

Proof. Let $\mathcal{A}$ be an adversary distinguishing between the two experiments $\mathrm{Exp}^{(0)\text{-ind-expl}'}_{\mathsf{DE},\mathcal{A}}(\lambda)$ and $\mathrm{Exp}^{(1)\text{-ind-expl}'}_{\mathsf{DE},\mathcal{A}}(\lambda)$ as above. Construct an adversary $\mathcal{B}$ breaking invertibility. Initially, $\mathcal{B}$ calls $\mathcal{A}$, obtains m^*, samples $(pk, sk) \leftarrow \mathsf{KGen}(1^\lambda)$ and outputs $m := (pk, m^*)$ to the experiment. In return, $\mathcal{B}$ receives (crs, r, y) from the experiment, where y is sampled using $\overline{\mathsf{Enc}}(pk, m^*)$ and r is either the randomness used by $\overline{\mathsf{Enc}}$ or is sampled via $\overline{\mathsf{Enc}}^{-1}(pk, m^*, y)$. $\mathcal{B}$ calls $\mathcal{A}$ on input of $((crs, pk), y, r)$. Hence,

$$\Pr\left[\mathrm{Exp}^{(b)\text{-ind-expl}'}_{\mathsf{DE},\mathcal{A}}(\lambda) = 1\right] = \Pr\left[\mathrm{Exp}^{(b)\text{-crs-inv}}_{(\mathsf{Setup}_{\mathsf{Enc}}, \overline{\mathsf{Enc}}, \overline{\mathsf{Enc}}^{-1}), \mathcal{B}}(\lambda) = 1\right]$$

for $b \in \{0, 1\}$. □



(a) Indistinguishability of explanation game using true randomness.

(b) Indistinguishability of explanation game using explained randomness.

Figure 18.10: Unwrapped static indistinguishability of explanation games used in the proof of indistinguishability of explanation of Theorem 18.6.

□

Whether $\mathsf{cISH}^{\mathsf{rand}}_{\approx_c}$ (without statistical closeness) implies deniable encryption remains open.

18.5 NON-COMMITTING ENCRYPTION

Non-committing encryption is a powerful notion which is known to imply adaptive MPC for well-formed functionalities [CFGN96; CLOS02].

On a high level, if $\mathsf{acPREH}^{\mathsf{rand}}_{\approx_c}$ is true, then any IND-CPA secure PKE scheme is a simulatable PKE scheme [DN00] (in the CRS model). The strategy of [CDMW09] translates to the CRS model (where inputs may be chosen depending on the CRS) directly.

DEFINITION 18.7 (Non-committing bit encryption in the global CRS model). A non-committing bit encryption (NCE) scheme in the global CRS model is a tuple of PPT algorithms $(\mathsf{Setup}, \mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Sim})$, such that $(\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ is a PKE scheme and the following distributions are computationally indistinguishable:

$$\left\{ \begin{array}{rl} crs & \leftarrow \mathsf{Setup}(1^\lambda) \\ (pk, sk) & \leftarrow \mathsf{KGen}(1^\lambda, crs; r_{\mathsf{KGen}}) \\ c & \leftarrow \mathsf{Enc}(crs, pk, b; r_{\mathsf{Enc}}) \end{array} : (crs, pk, c, r_{\mathsf{KGen}}, r_{\mathsf{Enc}}) \right\},$$

$$\left\{ \begin{array}{r} crs \leftarrow \mathsf{Setup}(1^\lambda) \\ (pk, c, r^0_{\mathsf{KGen}}, r^1_{\mathsf{KGen}}, r^0_{\mathsf{Enc}}, r^1_{\mathsf{Enc}}) \leftarrow \mathsf{Sim}(1^\lambda crs) \end{array} : (crs, pk, c, r^b_{\mathsf{KGen}}, r^b_{\mathsf{Enc}}) \right\}.$$

Following the lines of [CDMW09], if $\mathsf{acPREH}^{\mathsf{rand}}_{\approx_c}$ is true and there exists an IND-CPA secure PKE scheme, then there exists a non-committing encryption scheme.

THEOREM 18.7. *Let* $(\mathsf{KGen}', \mathsf{Enc}', \mathsf{Dec}')$ *be an IND-CPA secure PKE scheme for message space* $\{0,1\}^\lambda$. *If* $\mathsf{acPREH}^{\mathsf{rand}}_{\approx_c}$ *holds, then there exists a non-committing encryption scheme.*

$\mathsf{KGen}(1^\lambda, crs)$

$M_0, M_1 \leftarrow \{0,1\}^\lambda$

$T \leftarrow [4\lambda]$ s.t. $|T| = \lambda$

$$(pk_i, sk_i) \leftarrow \begin{cases} \mathsf{KGen}'(1^\lambda) & \text{if } i \in T \\ \overline{\mathsf{OGen}}(crs) & \text{otherwise} \end{cases}$$

$pk := (M_0, M_1, pk_1, \ldots, pk_{4\lambda})$

$sk := (T, (sk_i)_{i \in T})$

return (pk, sk)

(a) The key generation algorithm.

$\mathsf{Enc}(crs, pk, b)$

$S \leftarrow [4\lambda]$ s.t. $|S| = \lambda$

$$c_i \leftarrow \begin{cases} \mathsf{Enc}'(pk_i, M_b) & \text{if } i \in S \\ \overline{\mathsf{OEnc}}(crs, pk_i) & \text{otherwise} \end{cases}$$

return $c := (c_i)_{i \in [4\lambda]}$

(b) The encryption algorithm.

$\mathsf{Dec}(crs, sk, c)$

$J := \{\mathsf{Dec}'(sk_i, c_i) \mid i \in T\}$

if $M_0 \in J$ **then**

  return 0

else

  return 1

(c) The decryption algorithm.

Figure 18.11: Non-committing encryption scheme in the global CRS model based on [CDMW09].

Proof sketch. Let OGen be the algorithm which on input of 1^λ calls $\mathsf{KGen}'(1^\lambda)$ and outputs only $(pk, \perp)$. Further let OEnc be the algorithm which on input of $(1^\lambda, pk)$ samples $m \leftarrow \{0,1\}^\lambda$ and outputs $c \leftarrow \mathsf{Enc}'(1^\lambda, pk, m)$. If $\mathsf{acPREH}^{\mathsf{rand}}_{\approx_c}$ is true, then there exist alternative and inverse samplers for OGen and OEnc, denoted by $(\overline{\mathsf{OGen}}, \overline{\mathsf{OGen}}^{-1})$ and $(\overline{\mathsf{OEnc}}, \overline{\mathsf{OEnc}}^{-1})$, respectively. Note that the alternative sampler and the inverse sampler need access to the CRS. If the setup algorithm Setup is trivial (i. e., outputs the empty string), this corresponds to the notion of simulatable encryption due to [DN00] and yields non-committing encryption directly due to [CDMW09]. Figures 18.11 and 18.12 show the construction of non-committing encryption in the global CRS model. Note that assuming *adaptive* $\mathsf{acPREH}^{\mathsf{rand}}_{\approx_c}$ is necessary since the inputs to the sampler OEnc are sampled via $\mathsf{OGen}(crs)$ during the simulation and, hence, depend on the CRS. The indistinguishability between the real and the simulated distribution follows the same ideas as in [CDMW09]. □

18.6 SUPER-POLYNOMIAL ENCODING

Extremely lossy functions due to [Zha16] are functions which can be set up in two computationally indistinguishable modes – an injective mode and a extremely lossy mode, where the range of the function is merely polynomial, see Definition 4.2 in Section 4.2. A slight relaxation of this notion is what we call very lossy functions (VLFs). The difference to ELFs is that we require

Sim(crs)

$M_0, M_1 \leftarrow \{0,1\}^\lambda$

$S_0, T_0 \leftarrow [4\lambda]$ s.t. $|S_0| = |T_0| = \lambda$

$S_1, T_1 \leftarrow [4\lambda] \setminus (S_0 \cup T_0)$ s.t. $|S_0 \cap T_0| = |S_1 \cap T_1|$

$$(pk_i, sk_i) \leftarrow \begin{cases} \mathsf{KGen}'(1^\lambda; r^{(i)}_{\mathsf{KGen}'}) & \text{if } i \in T_0 \cup S_0 \cup T_1 \cup S_1 \\ \overline{\mathsf{OGen}}(crs; r^{(i)}_{\mathsf{OGen}}) & \text{otherwise} \end{cases}$$

$$c_i \leftarrow \begin{cases} \mathsf{Enc}'(pk_i, M_0; r^{(i)}_{\mathsf{Enc}'}) & \text{if } i \in S_0 \\ \mathsf{Enc}'(pk_i, M_1; r^{(i)}_{\mathsf{Enc}'}) & \text{if } i \in S_1 \\ \overline{\mathsf{OEnc}}(crs, pk_i; r^{(i)}_{\mathsf{OEnc}}) & \text{otherwise} \end{cases}$$

define $r^b_{\mathsf{KGen}} := (T_b, (u^{(b,i)}_{\mathsf{KGen}'})_{i\in[4\lambda]})$ and $r^b_{\mathsf{Enc}} := (S_b, (u^{(b,i)}_{\mathsf{Enc}'})_{i\in[4\lambda]})$

$$u^{(b,i)}_{\mathsf{KGen}'} \leftarrow \begin{cases} r^{(i)}_{\mathsf{KGen}'} & \text{if } i \in T_b \\ \overline{\mathsf{OGen}}^{-1}(crs, pk_i) & \text{if } i \in (T_0 \cup T_1 \cup S_0 \cup S_1) \setminus T_b \\ r^{(i)}_{\mathsf{OGen}} & \text{otherwise} \end{cases}$$

$$u^{(b,i)}_{\mathsf{Enc}'} \leftarrow \begin{cases} r^{(i)}_{\mathsf{Enc}'} & \text{if } i \in S_b \\ \overline{\mathsf{OEnc}}^{-1}(crs, (pk_i, M_{1-b}), c_i) & \text{if } i \in S_{1-b} \\ r^{(i)}_{\mathsf{OEnc}} & \text{otherwise} \end{cases}$$

return $\left(pk := (M_0, M_1, (pk_i)_{i\in[4\lambda]}), c := (c_i)_{i\in[4\lambda]}, r^0_{\mathsf{KGen}}, r^1_{\mathsf{KGen}}, r^0_{\mathsf{Enc}}, r^1_{\mathsf{Enc}}\right)$

Figure 18.12: The simulator Sim for the non-committing encryption scheme in the global CRS model described in Figure 18.11. Sim is based on the simulator given in [CDMW09].

indistinguishability between functions with exponential range and *super-polynomial* range. The existence of these functions implies a relaxation of $\mathsf{cPREH}^{\mathsf{rand}}_{\approx_c}$. The definition of VLFs is inspired by the definition of bounded adversary ELFs from [Zha16].

DEFINITION 18.8 (Very lossy functions). A *very lossy function* (VLF) is a tuple of PPT algorithms $\mathsf{VLF} = (\mathsf{Gen}, \mathsf{Eval})$ and a computable function $N(M)$ (such that $\log N$ is polynomial in $\log M$) such that the following properties are satisfied.

- Gen on input of M, $r \in [M]$ and a flag $b \in \{\mathsf{inj}, \mathsf{lossy}\}$, outputs a function description $G\colon [M] \to [N]$.
- If $b = \mathsf{inj}$, G is injective with overwhelming probability (in $\log M$).
- For all $r \in [M]$, if $b = \mathsf{lossy}$, $|G([M])| \leqslant r$ with overwhelming probability (in $\log M$).
- $\mathsf{Eval}(G, \cdot)$ evaluates the function G in polynomial time. A very lossy function is required to satisfy the following property.

INDISTINGUISHABILITY. There exists a *super-polynomial* function q such that for all PPT adversaries $\mathcal{A}$ and any $r \in [q(\log M), M]$,

$$\left| \Pr\left[\mathcal{A}(\mathsf{VLF.Gen}(M, r, \mathsf{inj})) = 1\right] - \Pr\left[\mathcal{A}(\mathsf{VLF.Gen}(M, r, \mathsf{lossy})) = 1\right] \right|$$

is negligible.

As a shorthand notation, we often write $G(x)$ to denote $\mathsf{VLF.Eval}(G, x)$.

DEFINITION 18.9 (Strong regularity, [Zha16]). A very lossy function VLF is *strongly regular* if for all $r \in [M]$, with overwhelming probability over the choice of $G \leftarrow \mathsf{VLF.Gen}(M, r)$, the distribution $\{x \leftarrow [M] : G(x)\}$ is statistically close to the uniform distribution over $G([M])$.

A VLF is called strongly efficiently enumerable, if there exists a (potentially randomized) algorithm running in polynomial time in $\log M$ and r, which given $G \leftarrow \mathsf{VLF.Gen}(M, r, \mathsf{lossy})$ and r outputs a set $S \subseteq [N]$ such that with overwhelming probability (over the choice of G and the randomness of the algorithm), we have $S = G([M])$. If VLF is strongly regular, then it also is strongly efficiently enumerable, [Zha16]. Zhandry [Zha16] shows that strongly regular ELFs are implied by the exponential decisional Diffie-Hellman (DDH) assumption.

We note that in contrast to ELFs, for VLFs an adversary who learns r does not harm security.

Assuming the subexponential hardness of the decisional Diffie-Hellman (DDH) problem, the bounded adversary extremely lossy function instantiation from [Zha16] is a strongly regular very lossy function according to Definitions 18.8 and 18.9.

THEOREM 18.8. *If strongly regular very lossy functions exist, then* $\mathsf{acPREH}^{\mathsf{rand}}_{\approx_c}$ *with super-polynomial encoding is true.*

Proof. Let S be a PPT sampler with input space L. For $m \in L$, let $\ell_r(|m|)$ denote the polynomial which upper bounds the number of random bits $S(m)$ takes, i. e., the random tape of S is uniform over $\{0,1\}^{\ell_r} = [2^{\ell_r}]$. Further, let ξ be a super-polynomial function. The setup, the encoding and the decoding algorithms are defined in Figure 18.13. Let $\overline{S}(crs, m; r) := D_S(crs, m, r)$ and $\overline{S}^{-1}(crs, m, y) := E_S(crs, m, y)$ be the corresponding alternative and inverse sampler as in the proof of Theorem 16.1. We prove the equivalent properties closeness and invertibility.

$\mathsf{Setup}(1^\lambda, 2^{\ell_r})$

$G \leftarrow \mathsf{VLF.Gen}(2^{\ell_r}, \xi, \mathsf{lossy})$

return $crs := G$

(a) The setup algorithm.

$D_S(crs, m, r)$

$y \leftarrow S(m; G(r))$

return y

(b) The decoding algorithm.

$E_S(crs, m, y)$

$\mathcal{R} := \varnothing$

for $r' \in \mathit{image}(G)$ **do**

  if $S(m; G(r')) = y$ **then**

    $\mathcal{R} := \mathcal{R} \cup r'$

$r \leftarrow \mathcal{R}$

return r

(c) The encoding algorithm.

Figure 18.13: Description of the pseudorandom encoding scheme with super-polynomial time encoding algorithm.

INVERTIBILITY. The encoding algorithm produces perfectly distributed inverse sampled random tapes. Hence, adaptive invertibility follows.

CLOSENESS. We start from the game $\mathsf{Exp}^{(b)\text{-a-crs-close}}_{(\mathsf{Setup},\overline{S},\overline{S}^{-1}),\mathcal{A}}(\lambda)$ and switch the VLF to injective mode $G \leftarrow \mathsf{VLF.Gen}(2^{\ell_r}, \xi, \mathsf{inj})$. This is computationally indistinguishable for the adversary. The inverse sampler will not work anymore, but since the adversary is polynomially bounded, he cannot call the inverse sampler anyway. Strong regularity implies that for $G \leftarrow \mathsf{VLF.Gen}(2^{\ell_r}, \xi, \mathsf{inj})$ the distribution $\{x \leftarrow [2^{\ell_r}] : G(x)\}$ is statistically close to uniform distribution over $G([2^{\ell_r}])$. Hence, $S(m; G(r))$ and $S(m; r)$ for uniform r from $[2^{\ell_r}]$ are statistically close. Then, the view of the adversary is identical regardless if we started from $\mathsf{Exp}^{(0)\text{-a-crs-close}}_{(\mathsf{Setup},\overline{S},\overline{S}^{-1}),\mathcal{A}}(\lambda)$ or $\mathsf{Exp}^{(1)\text{-a-crs-close}}_{(\mathsf{Setup},\overline{S},\overline{S}^{-1}),\mathcal{A}}(\lambda)$. □

Unfortunately, $\mathsf{acPREH}^{\mathsf{rand}}_{\approx_c}$ with super-polynomial encoding algorithm (or, equivalently $\mathsf{acISH}^{\mathsf{rand}}_{\approx_c}$ with super-polynomial inverse sampler) does not suffice to imply adaptive MPC even if the simulator is allowed to run in super-polynomial time, [Pas03]. This is because plugging $\mathsf{cPREH}^{\mathsf{rand}}_{\approx_c}$ with super-polynomial encoding algorithm into the proof of Theorem 18.2, the game hop from $\mathbf{G}_2$ to $\mathbf{G}_3$ cannot be made, since the simulation of these games requires super-polynomial time and, hence, a reduction to closeness against PPT adversaries is not possible. We do, however, obtain a non-standard notion of adaptive MPC with super-polynomial simulation, namely for any functionality $\mathcal{F}$ we are able to adaptively realize a functionality $\overline{\mathcal{F}}$ which produces a computationally (against PPT adversaries) indistinguishable output distribution to the original functionality. We view it as an interesting problem to further study this notion of adaptive MPC.

OUTLOOK AND FURTHER DIRECTIONS

19 OUTLOOK AND FURTHER DIRECTIONS

Finally, we provide an outlook on further interesting research directions.

USE OUR DPIO FRAMEWORK IN MORE SETTINGS. In Part I and [ACH20], we identify fully homomorphic encryption and spooky encryption to fit our dpIO framework and thereby improve the corresponding security proofs removing the need for subexponentially secure IO. Later, Döttling and Nishimaki [DN18] utilize our framework to improve the security reduction of their novel notion of universal proxy re-encryption. It seems plausible that our framework can be used to improve the security reduction of further primitives which require probabilistic IO. For instance, the IND-CCA1-secure fully homomorphic encryption scheme of [CRRV17] could be a good starting point.

FIND MORE APPLICATIONS FOR THE ALGEBRAIC WRAPPER. Since its introduction in [FKL18], the algebraic group model (AGM) enjoys an increasing popularity [FPS20; BFL20; RS20; KLRX20; GT20]. In Part II and [AHK20], we demonstrate how to implement AGM-based proofs from [FKL18] and [FPS20] in the standard (group) model. Further, we identify properties used in some proofs which exceed the capabilities of the algebraic wrapper. It would be interesting to identify further instances, where the extraction properties provided by the algebraic wrapper suffice. This would transport the corresponding proofs into the standard (group) model while avoiding knowledge-type assumptions.

NOVEL CONSTRUCTIONS OF IO. In Part III which is based on [ACI+20], we obtain partial evidence that indistinguishability obfuscation (or at least an assumption which stands in conflict with extractable one-way functions with unbounded auxiliary input) is necessary for pseudorandom encoding schemes for all efficiently samplable distributions. Hence, we raised the question whether IO is indeed necessary for pseudorandom encodings.

Inspired by [ACI+20], Wee and Wichs [WW20] answer this question in the affirmative assuming the subexponential hardness of the learning-with-errors (LWE) problem. This opens up new opportunities towards novel constructions of indistinguishability obfuscation. On the one hand, one can seek to improve the construction of [WW20]. On the other hand, the question whether IO is necessary for pseudorandom encoding schemes remains open relative to other hardness assumptions such as the LPN assumption or assumptions in cryptographic groups. We believe this will give rise to some new roads towards plausible IO candidates.

EXTEND COVERT COMPUTATION – HOW FAR CAN WE GO? Our notion of pseudorandom encodings from Part III has connections to variants of steganography and covert computation [ACI+20]. Covert computation, introduced in [vHL05; CGOS07], allows to hide the very fact that a multi-party

protocol is being executed and only reveals this fact (together with the protocol output) if an admissibility function is satisfied and all present parties actually participated. This admissibility function models that all participating parties are satisfied with the outcome of the protocol.

If a covert multi-party protocol terminates successfully, all participating parties learn not only their output but also the fact that all other parties actively participated. A natural question is whether we can transcend this notion of multi-party computation to hide participation even after a successful run. Whether MPC with deniable participation is feasible is an intriguing question currently under investigation. As a first step we identify "anonymous (message) transfer" (AT) as an interesting special case. An AT protocol considers three players: one receiver, one sender and one non-participant who only outputs random noise. The goal of AT is to deliver a message from the sender to the receiver in a way that leaves the receiver unable to determine which of the two other parties was the sender. Whether such an anonymous transfer protocol can exist is a very interesting question.

This is ongoing work together with Geoffroy Couteau, Sven Maier and Rafael Pass.

BIBLIOGRAPHY

[ABR01] Michel Abdalla, Mihir Bellare, and Phillip Rogaway. "The Oracle Diffie-Hellman Assumptions and an Analysis of DHIES." In: *Topics in Cryptology – CT-RSA 2001*. Ed. by David Naccache. Vol. 2020. Lecture Notes in Computer Science. San Francisco, CA, USA: Springer, Heidelberg, Germany, Apr. 2001, pp. 143–158. DOI: 10.1007/3-540-45353-9_12 (cit. on p. 129).

[Agr19] Shweta Agrawal. "Indistinguishability Obfuscation Without Multilinear Maps: New Methods for Bootstrapping and Instantiation." In: *Advances in Cryptology – EUROCRYPT 2019, Part I*. Ed. by Yuval Ishai and Vincent Rijmen. Vol. 11476. Lecture Notes in Computer Science. Darmstadt, Germany: Springer, Heidelberg, Germany, May 2019, pp. 191–225. DOI: 10.1007/978-3-030-17653-2_7 (cit. on pp. 2, 41).

[AP20] Shweta Agrawal and Alice Pellet-Mary. "Indistinguishability Obfuscation Without Maps: Attacks and Fixes for Noisy Linear FE." In: *Advances in Cryptology – EUROCRYPT 2020, Part I*. Ed. by Anne Canteaut and Yuval Ishai. Vol. 12105. Lecture Notes in Computer Science. Zagreb, Croatia: Springer, Heidelberg, Germany, May 2020, pp. 110–140. DOI: 10.1007/978-3-030-45721-1_5 (cit. on pp. 2, 41).

[Ajt96] Miklós Ajtai. "Generating Hard Instances of Lattice Problems (Extended Abstract)." In: *28th Annual ACM Symposium on Theory of Computing*. Philadephia, PA, USA: ACM Press, May 1996, pp. 99–108. DOI: 10.1145/237814.237838 (cit. on p. 2).

[AMP20] Navid Alamati, Hart Montgomery, and Sikhar Patranabis. *Stronger Multilinear Maps from Indistinguishability Obfuscation*. Cryptology ePrint Archive, Report 2020/610. https://eprint.iacr.org/2020/610. 2020 (cit. on pp. 91, 92, 110).

[AFH+16] Martin R. Albrecht, Pooya Farshim, Dennis Hofheinz, Enrique Larraia, and Kenneth G. Paterson. "Multilinear Maps from Obfuscation." In: *TCC 2016-A: 13th Theory of Cryptography Conference, Part I*. Ed. by Eyal Kushilevitz and Tal Malkin. Vol. 9562. Lecture Notes in Computer Science. Tel Aviv, Israel: Springer, Heidelberg, Germany, Jan. 2016, pp. 446–473. DOI: 10.1007/978-3-662-49096-9_19 (cit. on pp. 11, 90–92, 95, 110, 111, 115, 116, 120).

[ABG+13] Prabhanjan Ananth, Dan Boneh, Sanjam Garg, Amit Sahai, and Mark Zhandry. *Differing-Inputs Obfuscation and Applications*. Cryptology ePrint Archive, Report 2013/689. http://eprint.iacr.org/2013/689. 2013 (cit. on pp. 27, 41, 45).

[AJL+19] Prabhanjan Ananth, Aayush Jain, Huijia Lin, Christian Matt, and Amit Sahai. "Indistinguishability Obfuscation Without Multilinear Maps: New Paradigms via Low Degree Weak Pseudorandomness and Security Amplification." In: *Advances*

in Cryptology – CRYPTO 2019, Part III. Ed. by Alexandra Boldyreva and Daniele Micciancio. Vol. 11694. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 2019, pp. 284–332. DOI: 10.1007/978-3-030-26954-8_10 (cit. on pp. 2, 41).

[AJ15] Prabhanjan Ananth and Abhishek Jain. "Indistinguishability Obfuscation from Compact Functional Encryption." In: *Advances in Cryptology – CRYPTO 2015, Part I*. Ed. by Rosario Gennaro and Matthew J. B. Robshaw. Vol. 9215. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 2015, pp. 308–326. DOI: 10.1007/978-3-662-47989-6_15 (cit. on p. 41).

[AS17] Prabhanjan Ananth and Amit Sahai. "Projective Arithmetic Functional Encryption and Indistinguishability Obfuscation from Degree-5 Multilinear Maps." In: *Advances in Cryptology – EUROCRYPT 2017, Part I*. Ed. by Jean-Sébastien Coron and Jesper Buus Nielsen. Vol. 10210. Lecture Notes in Computer Science. Paris, France: Springer, Heidelberg, Germany, Apr. 2017, pp. 152–181. DOI: 10.1007/978-3-319-56620-7_6 (cit. on p. 41).

[ADGM17] Daniel Apon, Nico Döttling, Sanjam Garg, and Pratyay Mukherjee. "Cryptanalysis of Indistinguishability Obfuscations of Circuits over GGH13." In: *ICALP 2017: 44th International Colloquium on Automata, Languages and Programming*. Ed. by Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl. Vol. 80. LIPIcs. Warsaw, Poland: Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, July 2017, 38:1–38:16. DOI: 10.4230/LIPIcs.ICALP.2017.38 (cit. on p. 2).

[BMSZ16] Saikrishna Badrinarayanan, Eric Miles, Amit Sahai, and Mark Zhandry. "Post-zeroizing Obfuscation: New Mathematical Tools, and the Case of Evasive Circuits." In: *Advances in Cryptology – EUROCRYPT 2016, Part II*. Ed. by Marc Fischlin and Jean-Sébastien Coron. Vol. 9666. Lecture Notes in Computer Science. Vienna, Austria: Springer, Heidelberg, Germany, May 2016, pp. 764–791. DOI: 10.1007/978-3-662-49896-5_27 (cit. on pp. 2, 5).

[Bar01] Boaz Barak. "How to Go Beyond the Black-Box Simulation Barrier." In: *42nd Annual Symposium on Foundations of Computer Science*. Las Vegas, NV, USA: IEEE Computer Society Press, Oct. 2001, pp. 106–115. DOI: 10.1109/SFCS.2001.959885 (cit. on p. 32).

[BGI+01] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. "On the (Im)possibility of Obfuscating Programs." In: *Advances in Cryptology – CRYPTO 2001*. Ed. by Joe Kilian. Vol. 2139. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 2001, pp. 1–18. DOI: 10.1007/3-540-44647-8_1 (cit. on pp. 25, 41, 45).

[BGI+12] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. "On the (im)possibility of obfuscating programs." In: *J. ACM* 59.2 (2012), 6:1–6:48. DOI: 10.1145/2160158.2160159. URL: https://doi.org/10.1145/2160158.2160159 (cit. on pp. 2, 25, 27).

[BHHI10] Boaz Barak, Iftach Haitner, Dennis Hofheinz, and Yuval Ishai. "Bounded Key-Dependent Message Security." In: *Advances in Cryptology – EUROCRYPT 2010*. Ed. by Henri Gilbert. Vol. 6110. Lecture Notes in Computer Science. French Riviera: Springer, Heidelberg, Germany, May 2010, pp. 423–444. DOI: 10.1007/978-3-642-13190-5_22 (cit. on p. 46).

[BSW03] Boaz Barak, Ronen Shaltiel, and Avi Wigderson. "Computational Analogues of Entropy." In: *Approximation, Randomization, and Combinatorial Optimization: Algorithms and Techniques, 6th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems, APPROX 2003 and 7th International Workshop on Randomization and Approximation Techniques in Computer Science, RANDOM 2003, Princeton, NJ, USA, August 24-26, 2003, Proceedings*. Ed. by Sanjeev Arora, Klaus Jansen, José D. P. Rolim, and Amit Sahai. Vol. 2764. Lecture Notes in Computer Science. Springer, 2003, pp. 200–215. DOI: 10.1007/978-3-540-45198-3_18. URL: https://doi.org/10.1007/978-3-540-45198-3_18 (cit. on p. 190).

[BIJ+20] James Bartusek, Yuval Ishai, Aayush Jain, Fermi Ma, Amit Sahai, and Mark Zhandry. "Affine Determinant Programs: A Framework for Obfuscation and Witness Encryption." In: *ITCS 2020: 11th Innovations in Theoretical Computer Science Conference*. Ed. by Thomas Vidick. Vol. 151. Seattle, WA, USA: LIPIcs, Jan. 2020, 82:1–82:39. DOI: 10.4230/LIPIcs.ITCS.2020.82 (cit. on pp. 3, 5, 41).

[BFL20] Balthazar Bauer, Georg Fuchsbauer, and Julian Loss. "A Classification of Computational Assumptions in the Algebraic Group Model." In: *Advances in Cryptology – CRYPTO 2020, Part II*. Ed. by Daniele Micciancio and Thomas Ristenpart. Vol. 12171. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 2020, pp. 121–151. DOI: 10.1007/978-3-030-56880-1_5 (cit. on p. 245).

[Bea97] Donald Beaver. "Plug and Play Encryption." In: *Advances in Cryptology – CRYPTO'97*. Ed. by Burton S. Kaliski Jr. Vol. 1294. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 1997, pp. 75–89. DOI: 10.1007/BFb0052228 (cit. on pp. 8, 16).

[BH93] Donald Beaver and Stuart Haber. "Cryptographic Protocols Provably Secure Against Dynamic Adversaries." In: *Advances in Cryptology – EUROCRYPT'92*. Ed. by Rainer A. Rueppel. Vol. 658. Lecture Notes in Computer Science. Balatonfüred, Hungary: Springer, Heidelberg, Germany, May 1993, pp. 307–323. DOI: 10.1007/3-540-47555-9_26 (cit. on pp. 16, 154).

[BP04] Mihir Bellare and Adriana Palacio. "The Knowledge-of-Exponent Assumptions and 3-Round Zero-Knowledge Protocols." In: *Advances in Cryptology – CRYPTO 2004*. Ed. by Matthew Franklin. Vol. 3152. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 2004, pp. 273–289. DOI: 10.1007/978-3-540-28628-8_17 (cit. on pp. 32, 87).

[BR93] Mihir Bellare and Phillip Rogaway. "Random Oracles are Practical: A Paradigm for Designing Efficient Protocols." In: *ACM CCS 93: 1st Conference on Computer and Communications Security*. Ed. by Dorothy E. Denning, Raymond Pyle, Ravi Ganesan, Ravi S. Sandhu, and Victoria Ashby. Fairfax, Virginia, USA: ACM Press, Nov. 1993, pp. 62–73. DOI: 10.1145/168588.168596 (cit. on p. 10).

[BR95] Mihir Bellare and Phillip Rogaway. "Optimal Asymmetric Encryption." In: *Advances in Cryptology – EUROCRYPT'94*. Ed. by Alfredo De Santis. Vol. 950. Lecture Notes in Computer Science. Perugia, Italy: Springer, Heidelberg, Germany, May 1995, pp. 92–111. DOI: 10.1007/BFb0053428 (cit. on pp. 91, 129).

[BSW16] Mihir Bellare, Igors Stepanovs, and Brent Waters. "New Negative Results on Differing-Inputs Obfuscation." In: *Advances in Cryptology – EUROCRYPT 2016, Part II*. Ed. by Marc Fischlin and Jean-Sébastien Coron. Vol. 9666. Lecture Notes in Computer Science. Vienna, Austria: Springer, Heidelberg, Germany, May 2016, pp. 792–821. DOI: 10.1007/978-3-662-49896-5_28 (cit. on p. 45).

[BM92] Steven M. Bellovin and Michael Merritt. "Encrypted Key Exchange: Password-Based Protocols Secure against Dictionary Attacks." In: *1992 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, May 1992, pp. 72–84. DOI: 10.1109/RISP.1992.213269 (cit. on pp. 13, 147, 150).

[BL18a] Fabrice Benhamouda and Huijia Lin. "k-Round Multiparty Computation from k-Round Oblivious Transfer via Garbled Interactive Circuits." In: *Advances in Cryptology – EUROCRYPT 2018, Part II*. Ed. by Jesper Buus Nielsen and Vincent Rijmen. Vol. 10821. Lecture Notes in Computer Science. Tel Aviv, Israel: Springer, Heidelberg, Germany, Apr. 2018, pp. 500–532. DOI: 10.1007/978-3-319-78375-8_17 (cit. on pp. 3, 41).

[BL18b] Sebastian Berndt and Maciej Liskiewicz. "On the Gold Standard for Security of Universal Steganography." In: *Advances in Cryptology – EUROCRYPT 2018, Part I*. Ed. by Jesper Buus Nielsen and Vincent Rijmen. Vol. 10820. Lecture Notes in Computer Science. Tel Aviv, Israel: Springer, Heidelberg, Germany, Apr. 2018, pp. 29–60. DOI: 10.1007/978-3-319-78381-9_2 (cit. on p. 231).

[Bit17] Nir Bitansky. "Verifiable Random Functions from Non-interactive Witness-Indistinguishable Proofs." In: *TCC 2017: 15th Theory of Cryptography Conference, Part II*. Ed. by Yael Kalai

and Leonid Reyzin. Vol. 10678. Lecture Notes in Computer Science. Baltimore, MD, USA: Springer, Heidelberg, Germany, Nov. 2017, pp. 567–594. DOI: 10.1007/978-3-319-70503-3_19 (cit. on pp. 3, 41).

[BCC+17] Nir Bitansky, Ran Canetti, Alessandro Chiesa, Shafi Goldwasser, Huijia Lin, Aviad Rubinstein, and Eran Tromer. "The Hunting of the SNARK." In: *Journal of Cryptology* 30.4 (Oct. 2017), pp. 989–1066. DOI: 10.1007/s00145-016-9241-9 (cit. on p. 32).

[BCKP14] Nir Bitansky, Ran Canetti, Yael Tauman Kalai, and Omer Paneth. "On Virtual Grey Box Obfuscation for General Circuits." In: *Advances in Cryptology – CRYPTO 2014, Part II*. Ed. by Juan A. Garay and Rosario Gennaro. Vol. 8617. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 2014, pp. 108–125. DOI: 10.1007/978-3-662-44381-1_7 (cit. on p. 41).

[BCPR16] Nir Bitansky, Ran Canetti, Omer Paneth, and Alon Rosen. "On the Existence of Extractable One-Way Functions." In: *SIAM J. Comput.* 45.5 (2016), pp. 1910–1952. DOI: 10.1137/140975048. URL: https://doi.org/10.1137/140975048 (cit. on pp. 10, 31–33, 88, 89, 94, 109, 152, 153, 158, 200, 201, 211).

[BGL+15] Nir Bitansky, Sanjam Garg, Huijia Lin, Rafael Pass, and Sidharth Telang. "Succinct Randomized Encodings and their Applications." In: *47th Annual ACM Symposium on Theory of Computing*. Ed. by Rocco A. Servedio and Ronitt Rubinfeld. Portland, OR, USA: ACM Press, June 2015, pp. 439–448. DOI: 10.1145/2746539.2746574 (cit. on pp. 3, 80).

[BP13] Nir Bitansky and Omer Paneth. "On the impossibility of approximate obfuscation and applications to resettable cryptography." In: *45th Annual ACM Symposium on Theory of Computing*. Ed. by Dan Boneh, Tim Roughgarden, and Joan Feigenbaum. Palo Alto, CA, USA: ACM Press, June 2013, pp. 241–250. DOI: 10.1145/2488608.2488639 (cit. on p. 41).

[BP15a] Nir Bitansky and Omer Paneth. "ZAPs and Non-Interactive Witness Indistinguishability from Indistinguishability Obfuscation." In: *TCC 2015: 12th Theory of Cryptography Conference, Part II*. Ed. by Yevgeniy Dodis and Jesper Buus Nielsen. Vol. 9015. Lecture Notes in Computer Science. Warsaw, Poland: Springer, Heidelberg, Germany, Mar. 2015, pp. 401–427. DOI: 10.1007/978-3-662-46497-7_16 (cit. on pp. 3, 36, 61, 95).

[BPR15] Nir Bitansky, Omer Paneth, and Alon Rosen. "On the Cryptographic Hardness of Finding a Nash Equilibrium." In: *56th Annual Symposium on Foundations of Computer Science*. Ed. by Venkatesan Guruswami. Berkeley, CA, USA: IEEE Computer Society Press, Oct. 2015, pp. 1480–1498. DOI: 10.1109/FOCS.2015.94 (cit. on p. 3).

[BPW16] Nir Bitansky, Omer Paneth, and Daniel Wichs. "Perfect Structure on the Edge of Chaos - Trapdoor Permutations from Indistinguishability Obfuscation." In: *TCC 2016-A: 13th Theory of Cryptography Conference, Part I*. Ed. by Eyal Kushilevitz and Tal Malkin. Vol. 9562. Lecture Notes in Computer Science. Tel Aviv, Israel: Springer, Heidelberg, Germany, Jan. 2016, pp. 474–502. DOI: 10.1007/978-3-662-49096-9_20 (cit. on p. 3).

[BV15] Nir Bitansky and Vinod Vaikuntanathan. "Indistinguishability Obfuscation from Functional Encryption." In: *56th Annual Symposium on Foundations of Computer Science*. Ed. by Venkatesan Guruswami. Berkeley, CA, USA: IEEE Computer Society Press, Oct. 2015, pp. 171–190. DOI: 10.1109/FOCS.2015.20 (cit. on p. 41).

[BRS03] John Black, Phillip Rogaway, and Thomas Shrimpton. "Encryption-Scheme Security in the Presence of Key-Dependent Messages." In: *SAC 2002: 9th Annual International Workshop on Selected Areas in Cryptography*. Ed. by Kaisa Nyberg and Howard M. Heys. Vol. 2595. Lecture Notes in Computer Science. St. John's, Newfoundland, Canada: Springer, Heidelberg, Germany, Aug. 2003, pp. 62–75. DOI: 10.1007/3-540-36492-7_6 (cit. on p. 45).

[BFKL94] Avrim Blum, Merrick L. Furst, Michael J. Kearns, and Richard J. Lipton. "Cryptographic Primitives Based on Hard Learning Problems." In: *Advances in Cryptology – CRYPTO'93*. Ed. by Douglas R. Stinson. Vol. 773. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 1994, pp. 278–291. DOI: 10.1007/3-540-48329-2_24 (cit. on p. 2).

[BFM88] Manuel Blum, Paul Feldman, and Silvio Micali. "Non-Interactive Zero-Knowledge and Its Applications (Extended Abstract)." In: *20th Annual ACM Symposium on Theory of Computing*. Chicago, IL, USA: ACM Press, May 1988, pp. 103–112. DOI: 10.1145/62212.62222 (cit. on p. 35).

[BM82] Manuel Blum and Silvio Micali. "How to Generate Cryptographically Strong Sequences of Pseudo Random Bits." In: *23rd Annual Symposium on Foundations of Computer Science*. Chicago, Illinois: IEEE Computer Society Press, Nov. 1982, pp. 112–117. DOI: 10.1109/SFCS.1982.72 (cit. on p. 21).

[BB04a] Dan Boneh and Xavier Boyen. "Efficient Selective-ID Secure Identity Based Encryption Without Random Oracles." In: *Advances in Cryptology – EUROCRYPT 2004*. Ed. by Christian Cachin and Jan Camenisch. Vol. 3027. Lecture Notes in Computer Science. Interlaken, Switzerland: Springer, Heidelberg, Germany, May 2004, pp. 223–238. DOI: 10.1007/978-3-540-24676-3_14 (cit. on p. 43).

[BB04b] Dan Boneh and Xavier Boyen. "Secure Identity Based Encryption Without Random Oracles." In: *Advances in Cryptology – CRYPTO 2004*. Ed. by Matthew Franklin. Vol. 3152. Lecture Notes in Computer Science. Santa Barbara, CA, USA:

Springer, Heidelberg, Germany, Aug. 2004, pp. 443–459. DOI: 10.1007/978-3-540-28628-8_27 (cit. on p. 44).

[BF01] Dan Boneh and Matthew K. Franklin. "Identity-Based Encryption from the Weil Pairing." In: *Advances in Cryptology – CRYPTO 2001*. Ed. by Joe Kilian. Vol. 2139. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 2001, pp. 213–229. DOI: 10.1007/3-540-44647-8_13 (cit. on p. 8).

[BHHO08] Dan Boneh, Shai Halevi, Michael Hamburg, and Rafail Ostrovsky. "Circular-Secure Encryption from Decision Diffie-Hellman." In: *Advances in Cryptology – CRYPTO 2008*. Ed. by David Wagner. Vol. 5157. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 2008, pp. 108–125. DOI: 10.1007/978-3-540-85174-5_7 (cit. on p. 45).

[BL96] Dan Boneh and Richard J. Lipton. "Algorithms for Black-Box Fields and their Application to Cryptography (Extended Abstract)." In: *Advances in Cryptology – CRYPTO'96*. Ed. by Neal Koblitz. Vol. 1109. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 1996, pp. 283–297. DOI: 10.1007/3-540-68697-5_22 (cit. on p. 87).

[BLS04] Dan Boneh, Ben Lynn, and Hovav Shacham. "Short Signatures from the Weil Pairing." In: *Journal of Cryptology* 17.4 (Sept. 2004), pp. 297–319. DOI: 10.1007/s00145-004-0314-9 (cit. on pp. 12, 89).

[BV98] Dan Boneh and Ramarathnam Venkatesan. "Breaking RSA May Not Be Equivalent to Factoring." In: *Advances in Cryptology – EUROCRYPT'98*. Ed. by Kaisa Nyberg. Vol. 1403. Lecture Notes in Computer Science. Espoo, Finland: Springer, Heidelberg, Germany, May 1998, pp. 59–71. DOI: 10.1007/BFb0054117 (cit. on p. 87).

[BW13] Dan Boneh and Brent Waters. "Constrained Pseudorandom Functions and Their Applications." In: *Advances in Cryptology – ASIACRYPT 2013, Part II*. Ed. by Kazue Sako and Palash Sarkar. Vol. 8270. Lecture Notes in Computer Science. Bengalore, India: Springer, Heidelberg, Germany, Dec. 2013, pp. 280–300. DOI: 10.1007/978-3-642-42045-0_15 (cit. on pp. 23, 24).

[BZ14] Dan Boneh and Mark Zhandry. "Multiparty Key Exchange, Efficient Traitor Tracing, and More from Indistinguishability Obfuscation." In: *Advances in Cryptology – CRYPTO 2014, Part I*. Ed. by Juan A. Garay and Rosario Gennaro. Vol. 8616. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 2014, pp. 480–499. DOI: 10.1007/978-3-662-44371-2_27 (cit. on p. 3).

[BMN01] Colin Boyd, Paul Montague, and Khanh Quoc Nguyen. "Elliptic Curve Based Password Authenticated Key Exchange Protocols." In: *ACISP 01: 6th Australasian Conference on Information Security and Privacy*. Ed. by Vijay Varadharajan and Yi Mu. Vol. 2119. Lecture Notes in Computer Science. Sydney, NSW,

Australia: Springer, Heidelberg, Germany, July 2001, pp. 487–501. DOI: 10.1007/3-540-47719-5_38 (cit. on pp. 13, 150).

[Boy08] Xavier Boyen. "The Uber-Assumption Family." In: *Pairing-Based Cryptography - Pairing 2008, Second International Conference, Egham, UK, September 1-3, 2008. Proceedings*. 2008, pp. 39–56. DOI: 10.1007/978-3-540-85538-5_3. URL: https://doi.org/10.1007/978-3-540-85538-5_3 (cit. on p. 92).

[BCP14] Elette Boyle, Kai-Min Chung, and Rafael Pass. "On Extractability Obfuscation." In: *TCC 2014: 11th Theory of Cryptography Conference*. Ed. by Yehuda Lindell. Vol. 8349. Lecture Notes in Computer Science. San Diego, CA, USA: Springer, Heidelberg, Germany, Feb. 2014, pp. 52–73. DOI: 10.1007/978-3-642-54242-8_3 (cit. on pp. 27, 41).

[BCG+19a] Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, Peter Rindal, and Peter Scholl. "Efficient Two-Round OT Extension and Silent Non-Interactive Secure Computation." In: *ACM CCS 2019: 26th Conference on Computer and Communications Security*. Ed. by Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz. ACM Press, Nov. 2019, pp. 291–308. DOI: 10.1145/3319535.3354255 (cit. on p. 3).

[BCG+19b] Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, and Peter Scholl. "Efficient Pseudorandom Correlation Generators: Silent OT Extension and More." In: *Advances in Cryptology – CRYPTO 2019, Part III*. Ed. by Alexandra Boldyreva and Daniele Micciancio. Vol. 11694. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 2019, pp. 489–518. DOI: 10.1007/978-3-030-26954-8_16 (cit. on p. 3).

[BGI17] Elette Boyle, Niv Gilboa, and Yuval Ishai. "Group-Based Secure Computation: Optimizing Rounds, Communication, and Computation." In: *Advances in Cryptology – EUROCRYPT 2017, Part II*. Ed. by Jean-Sébastien Coron and Jesper Buus Nielsen. Vol. 10211. Lecture Notes in Computer Science. Paris, France: Springer, Heidelberg, Germany, Apr. 2017, pp. 163–193. DOI: 10.1007/978-3-319-56614-6_6 (cit. on p. 3).

[BGI14] Elette Boyle, Shafi Goldwasser, and Ioana Ivan. "Functional Signatures and Pseudorandom Functions." In: *PKC 2014: 17th International Conference on Theory and Practice of Public Key Cryptography*. Ed. by Hugo Krawczyk. Vol. 8383. Lecture Notes in Computer Science. Buenos Aires, Argentina: Springer, Heidelberg, Germany, Mar. 2014, pp. 501–519. DOI: 10.1007/978-3-642-54631-0_29 (cit. on pp. 23, 24).

[BP15b] Elette Boyle and Rafael Pass. "Limits of Extractability Assumptions with Distributional Auxiliary Input." In: *Advances in Cryptology – ASIACRYPT 2015, Part II*. Ed. by Tetsu Iwata and Jung Hee Cheon. Vol. 9453. Lecture Notes in Computer Science. Auckland, New Zealand: Springer, Heidelberg, Germany, Nov. 2015, pp. 236–261. DOI: 10.1007/978-3-662-48800-3_10 (cit. on p. 27).

[BDGM20a] Zvika Brakerski, Nico Döttling, Sanjam Garg, and Giulio Malavolta. "Candidate iO from Homomorphic Encryption Schemes." In: *Advances in Cryptology – EUROCRYPT 2020, Part I*. Ed. by Anne Canteaut and Yuval Ishai. Vol. 12105. Lecture Notes in Computer Science. Zagreb, Croatia: Springer, Heidelberg, Germany, May 2020, pp. 79–109. DOI: 10.1007/978-3-030-45721-1_4 (cit. on pp. 2, 5, 41, 158).

[BDGM20b] Zvika Brakerski, Nico Döttling, Sanjam Garg, and Giulio Malavolta. *Factoring and Pairings are not Necessary for iO: Circular-Secure LWE Suffices*. Cryptology ePrint Archive, Report 2020/1024. https://eprint.iacr.org/2020/1024. 2020 (cit. on pp. 2, 5, 41).

[BV11] Zvika Brakerski and Vinod Vaikuntanathan. "Fully Homomorphic Encryption from Ring-LWE and Security for Key Dependent Messages." In: *Advances in Cryptology – CRYPTO 2011*. Ed. by Phillip Rogaway. Vol. 6841. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 2011, pp. 505–524. DOI: 10.1007/978-3-642-22792-9_29 (cit. on p. 34).

[BV14a] Zvika Brakerski and Vinod Vaikuntanathan. "Efficient Fully Homomorphic Encryption from (Standard) LWE." In: *SIAM J. Comput.* 43.2 (2014), pp. 831–871. DOI: 10.1137/120868669. URL: https://doi.org/10.1137/120868669 (cit. on p. 34).

[BV14b] Zvika Brakerski and Vinod Vaikuntanathan. "Lattice-based FHE as secure as PKE." In: *ITCS 2014: 5th Conference on Innovations in Theoretical Computer Science*. Ed. by Moni Naor. Princeton, NJ, USA: Association for Computing Machinery, Jan. 2014, pp. 1–12. DOI: 10.1145/2554797.2554799 (cit. on p. 34).

[Can97] Ran Canetti. "Towards Realizing Random Oracles: Hash Functions That Hide All Partial Information." In: *Advances in Cryptology – CRYPTO'97*. Ed. by Burton S. Kaliski Jr. Vol. 1294. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 1997, pp. 455–469. DOI: 10.1007/BFb0052255 (cit. on p. 41).

[Can00] Ran Canetti. "Security and Composition of Multiparty Cryptographic Protocols." In: *Journal of Cryptology* 13.1 (Jan. 2000), pp. 143–202. DOI: 10.1007/s001459910006 (cit. on pp. 221, 222, 226).

[Can01] Ran Canetti. "Universally Composable Security: A New Paradigm for Cryptographic Protocols." In: *42nd Annual Symposium on Foundations of Computer Science*. Las Vegas, NV, USA: IEEE Computer Society Press, Oct. 2001, pp. 136–145. DOI: 10.1109/SFCS.2001.959888 (cit. on p. 223).

[CC17] Ran Canetti and Yilei Chen. "Constraint-Hiding Constrained PRFs for NC^1 from LWE." In: *Advances in Cryptology – EUROCRYPT 2017, Part I*. Ed. by Jean-Sébastien Coron and Jesper Buus Nielsen. Vol. 10210. Lecture Notes in Computer Science. Paris, France: Springer, Heidelberg, Germany, Apr. 2017,

pp. 446–476. DOI: 10.1007/978-3-319-56620-7_16 (cit. on p. 3).

[CCH+19] Ran Canetti, Yilei Chen, Justin Holmgren, Alex Lombardi, Guy N. Rothblum, Ron D. Rothblum, and Daniel Wichs. "Fiat-Shamir: from practice to theory." In: *51st Annual ACM Symposium on Theory of Computing*. Ed. by Moses Charikar and Edith Cohen. Phoenix, AZ, USA: ACM Press, June 2019, pp. 1082–1090. DOI: 10.1145/3313276.3316380 (cit. on p. 3).

[CCRR18] Ran Canetti, Yilei Chen, Leonid Reyzin, and Ron D. Rothblum. "Fiat-Shamir and Correlation Intractability from Strong KDM-Secure Encryption." In: *Advances in Cryptology – EUROCRYPT 2018, Part I*. Ed. by Jesper Buus Nielsen and Vincent Rijmen. Vol. 10820. Lecture Notes in Computer Science. Tel Aviv, Israel: Springer, Heidelberg, Germany, Apr. 2018, pp. 91–122. DOI: 10.1007/978-3-319-78381-9_4 (cit. on pp. 3, 41).

[CD08] Ran Canetti and Ronny Ramzi Dakdouk. "Extractable Perfectly One-Way Functions." In: *ICALP 2008: 35th International Colloquium on Automata, Languages and Programming, Part II*. Ed. by Luca Aceto, Ivan Damgård, Leslie Ann Goldberg, Magnús M. Halldórsson, Anna Ingólfsdóttir, and Igor Walukiewicz. Vol. 5126. Lecture Notes in Computer Science. Reykjavik, Iceland: Springer, Heidelberg, Germany, July 2008, pp. 449–460. DOI: 10.1007/978-3-540-70583-3_37 (cit. on p. 31).

[CD09] Ran Canetti and Ronny Ramzi Dakdouk. "Towards a Theory of Extractable Functions." In: *TCC 2009: 6th Theory of Cryptography Conference*. Ed. by Omer Reingold. Vol. 5444. Lecture Notes in Computer Science. Springer, Heidelberg, Germany, Mar. 2009, pp. 595–613. DOI: 10.1007/978-3-642-00457-5_35 (cit. on pp. 31, 32).

[CDPW07] Ran Canetti, Yevgeniy Dodis, Rafael Pass, and Shabsi Walfish. "Universally Composable Security with Global Setup." In: *TCC 2007: 4th Theory of Cryptography Conference*. Ed. by Salil P. Vadhan. Vol. 4392. Lecture Notes in Computer Science. Amsterdam, The Netherlands: Springer, Heidelberg, Germany, Feb. 2007, pp. 61–85. DOI: 10.1007/978-3-540-70936-7_4 (cit. on p. 223).

[CDNO97] Ran Canetti, Cynthia Dwork, Moni Naor, and Rafail Ostrovsky. "Deniable Encryption." In: *Advances in Cryptology – CRYPTO'97*. Ed. by Burton S. Kaliski Jr. Vol. 1294. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 1997, pp. 90–104. DOI: 10.1007/BFb0052229 (cit. on pp. 155, 233).

[CFGN96] Ran Canetti, Uriel Feige, Oded Goldreich, and Moni Naor. "Adaptively Secure Multi-Party Computation." In: *28th Annual ACM Symposium on Theory of Computing*. Philadephia, PA, USA: ACM Press, May 1996, pp. 639–648. DOI: 10.1145/237814.238015 (cit. on pp. 16, 17, 151, 154, 155, 159, 237).

[CL18] Ran Canetti and Amit Lichtenberg. "Certifying Trapdoor Permutations, Revisited." In: *TCC 2018: 16th Theory of Cryptography Conference, Part I*. Ed. by Amos Beimel and Stefan Dziembowski. Vol. 11239. Lecture Notes in Computer Science. Panaji, India: Springer, Heidelberg, Germany, Nov. 2018, pp. 476–506. DOI: 10.1007/978-3-030-03807-6_18 (cit. on p. 3).

[CLTV15] Ran Canetti, Huijia Lin, Stefano Tessaro, and Vinod Vaikuntanathan. "Obfuscation of Probabilistic Circuits and Applications." In: *TCC 2015: 12th Theory of Cryptography Conference, Part II*. Ed. by Yevgeniy Dodis and Jesper Buus Nielsen. Vol. 9015. Lecture Notes in Computer Science. Warsaw, Poland: Springer, Heidelberg, Germany, Mar. 2015, pp. 468–497. DOI: 10.1007/978-3-662-46497-7_19 (cit. on pp. iii, 3, 5, 8, 27–30, 34, 41–46, 56, 60, 67, 68, 73, 74, 80, 81, 89, 97, 99–105, 107).

[CLOS02] Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai. "Universally composable two-party and multi-party secure computation." In: *34th Annual ACM Symposium on Theory of Computing*. Montréal, Québec, Canada: ACM Press, May 2002, pp. 494–503. DOI: 10.1145/509907.509980 (cit. on pp. 16, 154, 222, 237).

[CPR17] Ran Canetti, Oxana Poburinnaya, and Mariana Raykova. "Optimal-Rate Non-Committing Encryption." In: *Advances in Cryptology – ASIACRYPT 2017, Part III*. Ed. by Tsuyoshi Takagi and Thomas Peyrin. Vol. 10626. Lecture Notes in Computer Science. Hong Kong, China: Springer, Heidelberg, Germany, Dec. 2017, pp. 212–241. DOI: 10.1007/978-3-319-70700-6_8 (cit. on pp. 150, 161, 171).

[CPV17] Ran Canetti, Oxana Poburinnaya, and Muthuramakrishnan Venkitasubramaniam. "Better Two-Round Adaptive Multi-party Computation." In: *PKC 2017: 20th International Conference on Theory and Practice of Public Key Cryptography, Part II*. Ed. by Serge Fehr. Vol. 10175. Lecture Notes in Computer Science. Amsterdam, The Netherlands: Springer, Heidelberg, Germany, Mar. 2017, pp. 396–427. DOI: 10.1007/978-3-662-54388-7_14 (cit. on pp. 16, 150, 161, 171).

[CRRV17] Ran Canetti, Srinivasan Raghuraman, Silas Richelson, and Vinod Vaikuntanathan. "Chosen-Ciphertext Secure Fully Homomorphic Encryption." In: *PKC 2017: 20th International Conference on Theory and Practice of Public Key Cryptography, Part II*. Ed. by Serge Fehr. Vol. 10175. Lecture Notes in Computer Science. Amsterdam, The Netherlands: Springer, Heidelberg, Germany, Mar. 2017, pp. 213–240. DOI: 10.1007/978-3-662-54388-7_8 (cit. on pp. 46, 245).

[Cha69] Gregory J. Chaitin. "On the Simplicity and Speed of Programs for Computing Infinite Sets of Natural Numbers." In: *J. ACM* 16.3 (1969), pp. 407–422. DOI: 10.1145/321526.321530. URL: https://doi.org/10.1145/321526.321530 (cit. on pp. 13, 147).

[CGOS07] Nishanth Chandran, Vipul Goyal, Rafail Ostrovsky, and Amit Sahai. "Covert Multi-Party Computation." In: *48th Annual Symposium on Foundations of Computer Science*. Providence, RI, USA: IEEE Computer Society Press, Oct. 2007, pp. 238–248. DOI: 10.1109/FOCS.2007.21 (cit. on pp. 13, 147, 155, 245).

[CKLM12] Melissa Chase, Markulf Kohlweiss, Anna Lysyanskaya, and Sarah Meiklejohn. "Malleable Proof Systems and Applications." In: *Advances in Cryptology – EUROCRYPT 2012*. Ed. by David Pointcheval and Thomas Johansson. Vol. 7237. Lecture Notes in Computer Science. Cambridge, UK: Springer, Heidelberg, Germany, Apr. 2012, pp. 281–300. DOI: 10.1007/978-3-642-29011-4_18 (cit. on p. 117).

[CGH17] Yilei Chen, Craig Gentry, and Shai Halevi. "Cryptanalyses of Candidate Branching Program Obfuscators." In: *Advances in Cryptology – EUROCRYPT 2017, Part III*. Ed. by Jean-Sébastien Coron and Jesper Buus Nielsen. Vol. 10212. Lecture Notes in Computer Science. Paris, France: Springer, Heidelberg, Germany, Apr. 2017, pp. 278–307. DOI: 10.1007/978-3-319-56617-7_10 (cit. on p. 2).

[CHVW19] Yilei Chen, Minki Hhan, Vinod Vaikuntanathan, and Hoeteck Wee. "Matrix PRFs: Constructions, Attacks, and Applications to Obfuscation." In: *TCC 2019: 17th Theory of Cryptography Conference, Part I*. Ed. by Dennis Hofheinz and Alon Rosen. Vol. 11891. Lecture Notes in Computer Science. Nuremberg, Germany: Springer, Heidelberg, Germany, Dec. 2019, pp. 55–80. DOI: 10.1007/978-3-030-36030-6_3 (cit. on pp. 2, 5, 41).

[CVW18] Yilei Chen, Vinod Vaikuntanathan, and Hoeteck Wee. "GGH15 Beyond Permutation Branching Programs: Proofs, Attacks, and Candidates." In: *Advances in Cryptology – CRYPTO 2018, Part II*. Ed. by Hovav Shacham and Alexandra Boldyreva. Vol. 10992. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 2018, pp. 577–607. DOI: 10.1007/978-3-319-96881-0_20 (cit. on pp. 2, 5, 41).

[CHL+15] Jung Hee Cheon, Kyoohyung Han, Changmin Lee, Hansol Ryu, and Damien Stehlé. "Cryptanalysis of the Multilinear Map over the Integers." In: *Advances in Cryptology – EUROCRYPT 2015, Part I*. Ed. by Elisabeth Oswald and Marc Fischlin. Vol. 9056. Lecture Notes in Computer Science. Sofia, Bulgaria: Springer, Heidelberg, Germany, Apr. 2015, pp. 3–12. DOI: 10.1007/978-3-662-46800-5_1 (cit. on p. 2).

[CDG+17] Chongwon Cho, Nico Döttling, Sanjam Garg, Divya Gupta, Peihan Miao, and Antigoni Polychroniadou. "Laconic Oblivious Transfer and Its Applications." In: *Advances in Cryptology – CRYPTO 2017, Part II*. Ed. by Jonathan Katz and Hovav Shacham. Vol. 10402. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 2017, pp. 33–65. DOI: 10.1007/978-3-319-63715-0_2 (cit. on p. 3).

[CDMW09] Seung Geol Choi, Dana Dachman-Soled, Tal Malkin, and Hoeteck Wee. "Improved Non-committing Encryption with Applications to Adaptively Secure Protocols." In: *Advances in Cryptology – ASIACRYPT 2009*. Ed. by Mitsuru Matsui. Vol. 5912. Lecture Notes in Computer Science. Tokyo, Japan: Springer, Heidelberg, Germany, Dec. 2009, pp. 287–302. DOI: 10.1007/978-3-642-10366-7_17 (cit. on pp. 153, 237–239).

[CLP15] Kai-Min Chung, Huijia Lin, and Rafael Pass. "Constant-Round Concurrent Zero-Knowledge from Indistinguishability Obfuscation." In: *Advances in Cryptology – CRYPTO 2015, Part I*. Ed. by Rosario Gennaro and Matthew J. B. Robshaw. Vol. 9215. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 2015, pp. 287–307. DOI: 10.1007/978-3-662-47989-6_14 (cit. on p. 3).

[CsW19] Ran Cohen, abhi shelat, and Daniel Wichs. "Adaptively Secure MPC with Sublinear Communication Complexity." In: *Advances in Cryptology – CRYPTO 2019, Part II*. Ed. by Alexandra Boldyreva and Daniele Micciancio. Vol. 11693. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 2019, pp. 30–60. DOI: 10.1007/978-3-030-26951-7_2 (cit. on pp. 4, 149, 153, 154, 187).

[CTL98a] Christian S. Collberg, Clark D. Thomborson, and Douglas Low. "Breaking Abstractions and Unstructuring Data Structures." In: *Proceedings of the 1998 International Conference on Computer Languages, ICCL 1998, Chicago, IL, USA, May 14-16, 1998*. IEEE Computer Society, 1998, pp. 28–38. DOI: 10.1109/ICCL.1998.674154. URL: https://doi.org/10.1109/ICCL.1998.674154 (cit. on p. 2).

[CTL98b] Christian S. Collberg, Clark D. Thomborson, and Douglas Low. "Manufacturing Cheap, Resilient, and Stealthy Opaque Constructs." In: *POPL '98, Proceedings of the 25th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, San Diego, CA, USA, January 19-21, 1998*. Ed. by David B. MacQueen and Luca Cardelli. ACM, 1998, pp. 184–196. DOI: 10.1145/268946.268962. URL: https://doi.org/10.1145/268946.268962 (cit. on p. 2).

[Coro00] Jean-Sébastien Coron. "On the Exact Security of Full Domain Hash." In: *Advances in Cryptology – CRYPTO 2000*. Ed. by Mihir Bellare. Vol. 1880. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 2000, pp. 229–235. DOI: 10.1007/3-540-44598-6_14 (cit. on p. 89).

[CLLT16] Jean-Sébastien Coron, Moon Sung Lee, Tancrède Lepoint, and Mehdi Tibouchi. "Cryptanalysis of GGH15 Multilinear Maps." In: *Advances in Cryptology – CRYPTO 2016, Part II*. Ed. by Matthew Robshaw and Jonathan Katz. Vol. 9815. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 2016, pp. 607–628. DOI: 10.1007/978-3-662-53008-5_21 (cit. on p. 2).

[CLLT17] Jean-Sébastien Coron, Moon Sung Lee, Tancrède Lepoint, and Mehdi Tibouchi. "Zeroizing Attacks on Indistinguishability Obfuscation over CLT13." In: *PKC 2017: 20th International Conference on Theory and Practice of Public Key Cryptography, Part I*. Ed. by Serge Fehr. Vol. 10174. Lecture Notes in Computer Science. Amsterdam, The Netherlands: Springer, Heidelberg, Germany, Mar. 2017, pp. 41–58. DOI: 10.1007/978-3-662-54365-8_3 (cit. on p. 2).

[CH19] Geoffroy Couteau and Dennis Hofheinz. "Designated-Verifier Pseudorandom Generators, and Their Applications." In: *Advances in Cryptology – EUROCRYPT 2019, Part II*. Ed. by Yuval Ishai and Vincent Rijmen. Vol. 11477. Lecture Notes in Computer Science. Darmstadt, Germany: Springer, Heidelberg, Germany, May 2019, pp. 562–592. DOI: 10.1007/978-3-030-17656-3_20 (cit. on p. 36).

[CS02] Ronald Cramer and Victor Shoup. "Universal Hash Proofs and a Paradigm for Adaptive Chosen Ciphertext Secure Public-Key Encryption." In: *Advances in Cryptology – EUROCRYPT 2002*. Ed. by Lars R. Knudsen. Vol. 2332. Lecture Notes in Computer Science. Amsterdam, The Netherlands: Springer, Heidelberg, Germany, Apr. 2002, pp. 45–64. DOI: 10.1007/3-540-46035-7_4 (cit. on p. 94).

[CS03] Ronald Cramer and Victor Shoup. "Design and Analysis of Practical Public-Key Encryption Schemes Secure against Adaptive Chosen Ciphertext Attack." In: *SIAM J. Comput.* 33.1 (2003), pp. 167–226. DOI: 10.1137/S0097539702403773. URL: https://doi.org/10.1137/S0097539702403773 (cit. on p. 129).

[Dac16] Dana Dachman-Soled. "Towards Non-Black-Box Separations of Public Key Encryption and One Way Function." In: *TCC 2016-B: 14th Theory of Cryptography Conference, Part II*. Ed. by Martin Hirt and Adam D. Smith. Vol. 9986. Lecture Notes in Computer Science. Beijing, China: Springer, Heidelberg, Germany, Oct. 2016, pp. 169–191. DOI: 10.1007/978-3-662-53644-5_7 (cit. on p. 33).

[DKR15] Dana Dachman-Soled, Jonathan Katz, and Vanishree Rao. "Adaptively Secure, Universally Composable, Multiparty Computation in Constant Rounds." In: *TCC 2015: 12th Theory of Cryptography Conference, Part II*. Ed. by Yevgeniy Dodis and Jesper Buus Nielsen. Vol. 9015. Lecture Notes in Computer Science. Warsaw, Poland: Springer, Heidelberg, Germany, Mar. 2015, pp. 586–613. DOI: 10.1007/978-3-662-46497-7_23 (cit. on pp. 16, 152–154, 187, 189, 200, 211–215, 221, 226, 233).

[Dam92] Ivan Damgård. "Towards Practical Public Key Systems Secure Against Chosen Ciphertext Attacks." In: *Advances in Cryptology – CRYPTO'91*. Ed. by Joan Feigenbaum. Vol. 576. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 1992, pp. 445–456. DOI: 10.1007/3-540-46766-1_36 (cit. on pp. 9, 30, 87, 88, 92).

[DJ01] Ivan Damgård and Mats Jurik. "A Generalisation, a Simplification and Some Applications of Paillier's Probabilistic Public-Key System." In: *PKC 2001: 4th International Workshop on Theory and Practice in Public Key Cryptography*. Ed. by Kwangjo Kim. Vol. 1992. Lecture Notes in Computer Science. Cheju Island, South Korea: Springer, Heidelberg, Germany, Feb. 2001, pp. 119–136. DOI: 10.1007/3-540-44586-2_9 (cit. on p. 97).

[DN00] Ivan Damgård and Jesper Buus Nielsen. "Improved Non-committing Encryption Schemes Based on a General Complexity Assumption." In: *Advances in Cryptology – CRYPTO 2000*. Ed. by Mihir Bellare. Vol. 1880. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 2000, pp. 432–450. DOI: 10.1007/3-540-44598-6_27 (cit. on pp. 17, 151, 159, 177, 237, 238).

[DPR16] Ivan Damgård, Antigoni Polychroniadou, and Vanishree Rao. "Adaptively Secure Multi-Party Computation from LWE (via Equivocal FHE)." In: *PKC 2016: 19th International Conference on Theory and Practice of Public Key Cryptography, Part II*. Ed. by Chen-Mou Cheng, Kai-Min Chung, Giuseppe Persiano, and Bo-Yin Yang. Vol. 9615. Lecture Notes in Computer Science. Taipei, Taiwan: Springer, Heidelberg, Germany, Mar. 2016, pp. 208–233. DOI: 10.1007/978-3-662-49387-8_9 (cit. on p. 16).

[Den02] Alexander W. Dent. "Adapting the Weaknesses of the Random Oracle Model to the Generic Group Model." In: *Advances in Cryptology – ASIACRYPT 2002*. Ed. by Yuliang Zheng. Vol. 2501. Lecture Notes in Computer Science. Queenstown, New Zealand: Springer, Heidelberg, Germany, Dec. 2002, pp. 100–109. DOI: 10.1007/3-540-36178-2_6 (cit. on pp. 9, 87).

[Den06] Alexander W. Dent. "The Cramer-Shoup Encryption Scheme Is Plaintext Aware in the Standard Model." In: *Advances in Cryptology – EUROCRYPT 2006*. Ed. by Serge Vaudenay. Vol. 4004. Lecture Notes in Computer Science. St. Petersburg, Russia: Springer, Heidelberg, Germany, May 2006, pp. 289–307. DOI: 10.1007/11761679_18 (cit. on p. 87).

[DH76] Whitfield Diffie and Martin E. Hellman. "New directions in cryptography." In: *IEEE Trans. Inf. Theory* 22.6 (1976), pp. 644–654. DOI: 10.1109/TIT.1976.1055638. URL: https://doi.org/10.1109/TIT.1976.1055638 (cit. on pp. 2, 8, 23).

[DHRW16] Yevgeniy Dodis, Shai Halevi, Ron D. Rothblum, and Daniel Wichs. "Spooky Encryption and Its Applications." In: *Advances in Cryptology – CRYPTO 2016, Part III*. Ed. by Matthew Robshaw and Jonathan Katz. Vol. 9816. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 2016, pp. 93–122. DOI: 10.1007/978-3-662-53015-3_4 (cit. on pp. 30, 41, 42, 46, 47).

[DORS08] Yevgeniy Dodis, Rafail Ostrovsky, Leonid Reyzin, and Adam D. Smith. "Fuzzy Extractors: How to Generate Strong Keys from Biometrics and Other Noisy Data." In: *SIAM J. Comput.*

38.1 (2008), pp. 97–139. DOI: 10.1137/060651380. URL: https://doi.org/10.1137/060651380 (cit. on pp. 20, 189).

[DRV12] Yevgeniy Dodis, Thomas Ristenpart, and Salil P. Vadhan. "Randomness Condensers for Efficiently Samplable, Seed-Dependent Sources." In: *TCC 2012: 9th Theory of Cryptography Conference*. Ed. by Ronald Cramer. Vol. 7194. Lecture Notes in Computer Science. Taormina, Sicily, Italy: Springer, Heidelberg, Germany, Mar. 2012, pp. 618–635. DOI: 10.1007/978-3-642-28914-9_35 (cit. on pp. 13, 147).

[DG17] Nico Döttling and Sanjam Garg. "Identity-Based Encryption from the Diffie-Hellman Assumption." In: *Advances in Cryptology – CRYPTO 2017, Part I*. Ed. by Jonathan Katz and Hovav Shacham. Vol. 10401. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 2017, pp. 537–569. DOI: 10.1007/978-3-319-63688-7_18 (cit. on p. 3).

[DGI+19] Nico Döttling, Sanjam Garg, Yuval Ishai, Giulio Malavolta, Tamer Mour, and Rafail Ostrovsky. "Trapdoor Hash Functions and Their Applications." In: *Advances in Cryptology – CRYPTO 2019, Part III*. Ed. by Alexandra Boldyreva and Daniele Micciancio. Vol. 11694. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 2019, pp. 3–32. DOI: 10.1007/978-3-030-26954-8_1 (cit. on p. 3).

[DN18] Nico Döttling and Ryo Nishimaki. *Universal Proxy Re-Encryption*. Cryptology ePrint Archive, Report 2018/840. https://eprint.iacr.org/2018/840. 2018 (cit. on pp. 8, 46, 245).

[ElG85] Taher ElGamal. "A public key cryptosystem and a signature scheme based on discrete logarithms." In: *IEEE Trans. Information Theory* 31.4 (1985), pp. 469–472. DOI: 10.1109/TIT.1985.1057074. URL: https://doi.org/10.1109/TIT.1985.1057074 (cit. on pp. 8, 33, 97, 129).

[EGL85] Shimon Even, Oded Goldreich, and Abraham Lempel. "A Randomized Protocol for Signing Contracts." In: *Commun. ACM* 28.6 (1985), pp. 637–647. DOI: 10.1145/3812.3818. URL: http://doi.acm.org/10.1145/3812.3818 (cit. on p. 159).

[FHHL18] Pooya Farshim, Julia Hesse, Dennis Hofheinz, and Enrique Larraia. "Graded Encoding Schemes from Obfuscation." In: *PKC 2018: 21st International Conference on Theory and Practice of Public Key Cryptography, Part II*. Ed. by Michel Abdalla and Ricardo Dahab. Vol. 10770. Lecture Notes in Computer Science. Rio de Janeiro, Brazil: Springer, Heidelberg, Germany, Mar. 2018, pp. 371–400. DOI: 10.1007/978-3-319-76581-5_13 (cit. on pp. 41, 91, 92, 110, 111, 115, 116).

[FLS90] Uriel Feige, Dror Lapidot, and Adi Shamir. "Multiple Non-Interactive Zero Knowledge Proofs Based on a Single Random String (Extended Abstract)." In: *31st Annual Symposium on Foundations of Computer Science*. St. Louis, MO, USA: IEEE

Computer Society Press, Oct. 1990, pp. 308–317. DOI: 10.1109/FSCS.1990.89549 (cit. on p. 36).

[FS90] Uriel Feige and Adi Shamir. "Witness Indistinguishable and Witness Hiding Protocols." In: *22nd Annual ACM Symposium on Theory of Computing*. Baltimore, MD, USA: ACM Press, May 1990, pp. 416–426. DOI: 10.1145/100216.100272 (cit. on p. 95).

[FS87] Amos Fiat and Adi Shamir. "How to Prove Yourself: Practical Solutions to Identification and Signature Problems." In: *Advances in Cryptology – CRYPTO'86*. Ed. by Andrew M. Odlyzko. Vol. 263. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 1987, pp. 186–194. DOI: 10.1007/3-540-47721-7_12 (cit. on p. 10).

[FGJ18] Nils Fleischhacker, Vipul Goyal, and Abhishek Jain. "On the Existence of Three Round Zero-Knowledge Proofs." In: *Advances in Cryptology – EUROCRYPT 2018, Part III*. Ed. by Jesper Buus Nielsen and Vincent Rijmen. Vol. 10822. Lecture Notes in Computer Science. Tel Aviv, Israel: Springer, Heidelberg, Germany, Apr. 2018, pp. 3–33. DOI: 10.1007/978-3-319-78372-7_1 (cit. on p. 32).

[FJS19] Nils Fleischhacker, Tibor Jager, and Dominique Schröder. "On Tight Security Proofs for Schnorr Signatures." In: *Journal of Cryptology* 32.2 (Apr. 2019), pp. 566–599. DOI: 10.1007/s00145-019-09311-5 (cit. on p. 89).

[Flo64] William B. Floyd. "Review of 'Information Theory and Coding' (Abramson, N.; 1963)." In: *IEEE Trans. Inf. Theory* 10.4 (1964), p. 392. DOI: 10.1109/TIT.1964.1053709. URL: https://doi.org/10.1109/TIT.1964.1053709 (cit. on p. 12).

[FJNT15] Tore Kasper Frederiksen, Thomas P. Jakobsen, Jesper Buus Nielsen, and Roberto Trifiletti. *TinyLEGO: An Interactive Garbling Scheme for Maliciously Secure Two-Party Computation*. Cryptology ePrint Archive, Report 2015/309. http://eprint.iacr.org/2015/309. 2015 (cit. on p. 3).

[FHPS13] Eduarda S. V. Freire, Dennis Hofheinz, Kenneth G. Paterson, and Christoph Striecks. "Programmable Hash Functions in the Multilinear Setting." In: *Advances in Cryptology – CRYPTO 2013, Part I*. Ed. by Ran Canetti and Juan A. Garay. Vol. 8042. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 2013, pp. 513–530. DOI: 10.1007/978-3-642-40041-4_28 (cit. on pp. 10, 90).

[FKL18] Georg Fuchsbauer, Eike Kiltz, and Julian Loss. "The Algebraic Group Model and its Applications." In: *Advances in Cryptology – CRYPTO 2018, Part II*. Ed. by Hovav Shacham and Alexandra Boldyreva. Vol. 10992. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 2018, pp. 33–62. DOI: 10.1007/978-3-319-96881-0_2 (cit. on pp. iii, 9, 10, 12, 87–89, 91, 93, 94, 111, 245).

[FPS20] Georg Fuchsbauer, Antoine Plouviez, and Yannick Seurin. "Blind Schnorr Signatures and Signed ElGamal Encryption in the Algebraic Group Model." In: *Advances in Cryptology – EUROCRYPT 2020, Part II*. Ed. by Anne Canteaut and Yuval Ishai. Vol. 12106. Lecture Notes in Computer Science. Zagreb, Croatia: Springer, Heidelberg, Germany, May 2020, pp. 63–95. DOI: 10.1007/978-3-030-45724-2_3 (cit. on pp. 12, 88, 89, 91, 92, 129–133, 139–141, 245).

[GGH13] Sanjam Garg, Craig Gentry, and Shai Halevi. "Candidate Multilinear Maps from Ideal Lattices." In: *Advances in Cryptology – EUROCRYPT 2013*. Ed. by Thomas Johansson and Phong Q. Nguyen. Vol. 7881. Lecture Notes in Computer Science. Athens, Greece: Springer, Heidelberg, Germany, May 2013, pp. 1–17. DOI: 10.1007/978-3-642-38348-9_1 (cit. on pp. 90, 109, 110).

[GGHR14] Sanjam Garg, Craig Gentry, Shai Halevi, and Mariana Raykova. "Two-Round Secure MPC from Indistinguishability Obfuscation." In: *TCC 2014: 11th Theory of Cryptography Conference*. Ed. by Yehuda Lindell. Vol. 8349. Lecture Notes in Computer Science. San Diego, CA, USA: Springer, Heidelberg, Germany, Feb. 2014, pp. 74–94. DOI: 10.1007/978-3-642-54242-8_4 (cit. on pp. 3, 41).

[GGH+13] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. "Candidate Indistinguishability Obfuscation and Functional Encryption for all Circuits." In: *54th Annual Symposium on Foundations of Computer Science*. Berkeley, CA, USA: IEEE Computer Society Press, Oct. 2013, pp. 40–49. DOI: 10.1109/FOCS.2013.13 (cit. on pp. 2, 3, 25, 26, 41).

[GGHW17] Sanjam Garg, Craig Gentry, Shai Halevi, and Daniel Wichs. "On the Implausibility of Differing-Inputs Obfuscation and Extractable Witness Encryption with Auxiliary Input." In: *Algorithmica* 79.4 (2017), pp. 1353–1373. DOI: 10.1007/s00453-017-0276-6. URL: https://doi.org/10.1007/s00453-017-0276-6 (cit. on pp. 27, 28, 45, 60).

[GMM+16] Sanjam Garg, Eric Miles, Pratyay Mukherjee, Amit Sahai, Akshayaram Srinivasan, and Mark Zhandry. "Secure Obfuscation in a Weak Multilinear Map Model." In: *TCC 2016-B: 14th Theory of Cryptography Conference, Part II*. Ed. by Martin Hirt and Adam D. Smith. Vol. 9986. Lecture Notes in Computer Science. Beijing, China: Springer, Heidelberg, Germany, Oct. 2016, pp. 241–268. DOI: 10.1007/978-3-662-53644-5_10 (cit. on pp. 2, 5).

[GPS16] Sanjam Garg, Omkant Pandey, and Akshayaram Srinivasan. "Revisiting the Cryptographic Hardness of Finding a Nash Equilibrium." In: *Advances in Cryptology – CRYPTO 2016, Part II*. Ed. by Matthew Robshaw and Jonathan Katz. Vol. 9815. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 2016, pp. 579–604. DOI: 10.1007/978-3-662-53008-5_20 (cit. on pp. 3, 5, 42).

[GPSZ17] Sanjam Garg, Omkant Pandey, Akshayaram Srinivasan, and Mark Zhandry. "Breaking the Sub-Exponential Barrier in Obfustopia." In: *Advances in Cryptology – EUROCRYPT 2017, Part III*. Ed. by Jean-Sébastien Coron and Jesper Buus Nielsen. Vol. 10212. Lecture Notes in Computer Science. Paris, France: Springer, Heidelberg, Germany, Apr. 2017, pp. 156–181. DOI: 10.1007/978-3-319-56617-7_6 (cit. on pp. 5, 42, 44).

[GS12] Sanjam Garg and Amit Sahai. "Adaptively Secure Multi-Party Computation with Dishonest Majority." In: *Advances in Cryptology – CRYPTO 2012*. Ed. by Reihaneh Safavi-Naini and Ran Canetti. Vol. 7417. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 2012, pp. 105–123. DOI: 10.1007/978-3-642-32009-5_8 (cit. on pp. 16, 154).

[GS16] Sanjam Garg and Akshayaram Srinivasan. "Single-Key to Multi-Key Functional Encryption with Polynomial Loss." In: *TCC 2016-B: 14th Theory of Cryptography Conference, Part II*. Ed. by Martin Hirt and Adam D. Smith. Vol. 9986. Lecture Notes in Computer Science. Beijing, China: Springer, Heidelberg, Germany, Oct. 2016, pp. 419–442. DOI: 10.1007/978-3-662-53644-5_16 (cit. on p. 42).

[GS18] Sanjam Garg and Akshayaram Srinivasan. "Two-Round Multiparty Secure Computation from Minimal Assumptions." In: *Advances in Cryptology – EUROCRYPT 2018, Part II*. Ed. by Jesper Buus Nielsen and Vincent Rijmen. Vol. 10821. Lecture Notes in Computer Science. Tel Aviv, Israel: Springer, Heidelberg, Germany, Apr. 2018, pp. 468–499. DOI: 10.1007/978-3-319-78375-8_16 (cit. on pp. 3, 41).

[GJLS20] Romain Gay, Aayush Jain, Huijia Lin, and Amit Sahai. *Indistinguishability Obfuscation from Simple-to-State Hard Problems: New Assumptions, New Techniques, and Simplification*. Cryptology ePrint Archive, Report 2020/764. https://eprint.iacr.org/2020/764. 2020 (cit. on pp. 2, 5, 41).

[GP20] Romain Gay and Rafael Pass. *Indistinguishability Obfuscation from Circular Security*. Cryptology ePrint Archive, Report 2020/1010. https://eprint.iacr.org/2020/1010. 2020 (cit. on pp. 2, 5, 41).

[Gen09] Craig Gentry. "Fully homomorphic encryption using ideal lattices." In: *41st Annual ACM Symposium on Theory of Computing*. Ed. by Michael Mitzenmacher. Bethesda, MD, USA: ACM Press, May 2009, pp. 169–178. DOI: 10.1145/1536414.1536440 (cit. on pp. 34, 73).

[GGH15] Craig Gentry, Sergey Gorbunov, and Shai Halevi. "Graph-Induced Multilinear Maps from Lattices." In: *TCC 2015: 12th Theory of Cryptography Conference, Part II*. Ed. by Yevgeniy Dodis and Jesper Buus Nielsen. Vol. 9015. Lecture Notes in Computer Science. Warsaw, Poland: Springer, Heidelberg, Germany, Mar. 2015, pp. 498–527. DOI: 10.1007/978-3-662-46497-7_20 (cit. on pp. 2, 3, 5).

[GJK18] Craig Gentry, Charanjit S. Jutla, and Daniel Kane. *Obfuscation Using Tensor Products*. Cryptology ePrint Archive, Report 2018/756. https://eprint.iacr.org/2018/756. 2018 (cit. on pp. 3, 5, 41).

[GKM+00] Yael Gertner, Sampath Kannan, Tal Malkin, Omer Reingold, and Mahesh Viswanathan. "The Relationship between Public Key Encryption and Oblivious Transfer." In: *41st Annual Symposium on Foundations of Computer Science*. Redondo Beach, CA, USA: IEEE Computer Society Press, Nov. 2000, pp. 325–335. DOI: 10.1109/SFCS.2000.892121 (cit. on pp. 17, 151, 159).

[GT20] Ashrujit Ghoshal and Stefano Tessaro. *Tight State-Restoration Soundness in the Algebraic Group Model*. Cryptology ePrint Archive, Report 2020/1351. https://eprint.iacr.org/2020/1351. 2020 (cit. on p. 245).

[GM59] E. N. Gilbert and E. F. Moore. "Variable-length binary encodings." In: *The Bell System Technical Journal* 38.4 (1959), pp. 933–967. DOI: 10.1002/j.1538-7305.1959.tb01583.x (cit. on p. 12).

[GS85] Andrew V. Goldberg and Michael Sipser. "Compression and Ranking." In: *17th Annual ACM Symposium on Theory of Computing*. Providence, RI, USA: ACM Press, May 1985, pp. 440–448. DOI: 10.1145/22145.22194 (cit. on pp. 12, 13, 147, 150, 156).

[GGM84a] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. "How to Construct Random Functions (Extended Abstract)." In: *25th Annual Symposium on Foundations of Computer Science*. Singer Island, Florida: IEEE Computer Society Press, Oct. 1984, pp. 464–479. DOI: 10.1109/SFCS.1984.715949 (cit. on p. 61).

[GGM84b] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. "On the Cryptographic Applications of Random Functions." In: *Advances in Cryptology – CRYPTO'84*. Ed. by G. R. Blakley and David Chaum. Vol. 196. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 1984, pp. 276–288 (cit. on p. 10).

[GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. "How to construct random functions." In: *J. ACM* 33.4 (1986), pp. 792–807. DOI: 10.1145/6490.6503. URL: https://doi.org/10.1145/6490.6503 (cit. on pp. 10, 23, 24).

[GK96] Oded Goldreich and Hugo Krawczyk. "On the Composition of Zero-Knowledge Proof Systems." In: *SIAM J. Comput.* 25.1 (1996), pp. 169–192. DOI: 10.1137/S0097539791220688. URL: https://doi.org/10.1137/S0097539791220688 (cit. on p. 32).

[GMW86] Oded Goldreich, Silvio Micali, and Avi Wigderson. "Proofs that Yield Nothing But their Validity and a Methodology of Cryptographic Protocol Design (Extended Abstract)." In: *27th Annual Symposium on Foundations of Computer Science*. Toronto, Ontario, Canada: IEEE Computer Society Press, Oct. 1986, pp. 174–187. DOI: 10.1109/SFCS.1986.47 (cit. on p. 36).

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. "How to Play any Mental Game or A Completeness Theorem for Protocols with Honest Majority." In: *19th Annual ACM Symposium on Theory of Computing*. Ed. by Alfred Aho. New York City, NY, USA: ACM Press, May 1987, pp. 218–229. DOI: 10.1145/28395.28420 (cit. on pp. 16, 221).

[GO94] Oded Goldreich and Yair Oren. "Definitions and Properties of Zero-Knowledge Proof Systems." In: *Journal of Cryptology* 7.1 (Dec. 1994), pp. 1–32. DOI: 10.1007/BF00195207 (cit. on p. 35).

[GGG+14] Shafi Goldwasser, S. Dov Gordon, Vipul Goyal, Abhishek Jain, Jonathan Katz, Feng-Hao Liu, Amit Sahai, Elaine Shi, and Hong-Sheng Zhou. "Multi-input Functional Encryption." In: *Advances in Cryptology – EUROCRYPT 2014*. Ed. by Phong Q. Nguyen and Elisabeth Oswald. Vol. 8441. Lecture Notes in Computer Science. Copenhagen, Denmark: Springer, Heidelberg, Germany, May 2014, pp. 578–602. DOI: 10.1007/978-3-642-55220-5_32 (cit. on p. 3).

[GK05] Shafi Goldwasser and Yael Tauman Kalai. "On the Impossibility of Obfuscation with Auxiliary Input." In: *46th Annual Symposium on Foundations of Computer Science*. Pittsburgh, PA, USA: IEEE Computer Society Press, Oct. 2005, pp. 553–562. DOI: 10.1109/SFCS.2005.60 (cit. on p. 41).

[GL91] Shafi Goldwasser and Leonid A. Levin. "Fair Computation of General Functions in Presence of Immoral Majority." In: *Advances in Cryptology – CRYPTO'90*. Ed. by Alfred J. Menezes and Scott A. Vanstone. Vol. 537. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 1991, pp. 77–93. DOI: 10.1007/3-540-38424-3_6 (cit. on p. 224).

[GM82] Shafi Goldwasser and Silvio Micali. "Probabilistic Encryption and How to Play Mental Poker Keeping Secret All Partial Information." In: *14th Annual ACM Symposium on Theory of Computing*. San Francisco, CA, USA: ACM Press, May 1982, pp. 365–377. DOI: 10.1145/800070.802212 (cit. on p. 21).

[GM84] Shafi Goldwasser and Silvio Micali. "Probabilistic Encryption." In: *J. Comput. Syst. Sci.* 28.2 (1984), pp. 270–299. DOI: 10.1016/0022-0000(84)90070-9. URL: https://doi.org/10.1016/0022-0000(84)90070-9 (cit. on pp. 33, 97).

[GMR85] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. "The Knowledge Complexity of Interactive Proof-Systems (Extended Abstract)." In: *17th Annual ACM Symposium on Theory of Computing*. Providence, RI, USA: ACM Press, May 1985, pp. 291–304. DOI: 10.1145/22145.22178 (cit. on p. 35).

[GR07] Shafi Goldwasser and Guy N. Rothblum. "On Best-Possible Obfuscation." In: *TCC 2007: 4th Theory of Cryptography Conference*. Ed. by Salil P. Vadhan. Vol. 4392. Lecture Notes in Computer Science. Amsterdam, The Netherlands: Springer, Heidelberg, Germany, Feb. 2007, pp. 194–213. DOI: 10.1007/978-3-540-70936-7_11 (cit. on pp. 2, 25, 41).

[GHKW17] Rishab Goyal, Susan Hohenberger, Venkata Koppula, and Brent Waters. "A Generic Approach to Constructing and Proving Verifiable Random Functions." In: *TCC 2017: 15th Theory of Cryptography Conference, Part II*. Ed. by Yael Kalai and Leonid Reyzin. Vol. 10678. Lecture Notes in Computer Science. Baltimore, MD, USA: Springer, Heidelberg, Germany, Nov. 2017, pp. 537–566. DOI: 10.1007/978-3-319-70503-3_18 (cit. on pp. 3, 41).

[GKW17] Rishab Goyal, Venkata Koppula, and Brent Waters. "Separating Semantic and Circular Security for Symmetric-Key Bit Encryption from the Learning with Errors Assumption." In: *Advances in Cryptology – EUROCRYPT 2017, Part II*. Ed. by Jean-Sébastien Coron and Jesper Buus Nielsen. Vol. 10211. Lecture Notes in Computer Science. Paris, France: Springer, Heidelberg, Germany, Apr. 2017, pp. 528–557. DOI: 10.1007/978-3-319-56614-6_18 (cit. on p. 3).

[GOS06] Jens Groth, Rafail Ostrovsky, and Amit Sahai. "Perfect Non-interactive Zero Knowledge for NP." In: *Advances in Cryptology – EUROCRYPT 2006*. Ed. by Serge Vaudenay. Vol. 4004. Lecture Notes in Computer Science. St. Petersburg, Russia: Springer, Heidelberg, Germany, May 2006, pp. 339–358. DOI: 10.1007/11761679_21 (cit. on pp. 35, 36).

[GOS12] Jens Groth, Rafail Ostrovsky, and Amit Sahai. "New Techniques for Noninteractive Zero-Knowledge." In: *J. ACM* 59.3 (2012), 11:1–11:35. DOI: 10.1145/2220357.2220358. URL: https://doi.org/10.1145/2220357.2220358 (cit. on p. 8).

[GS08] Jens Groth and Amit Sahai. "Efficient Non-interactive Proof Systems for Bilinear Groups." In: *Advances in Cryptology – EUROCRYPT 2008*. Ed. by Nigel P. Smart. Vol. 4965. Lecture Notes in Computer Science. Istanbul, Turkey: Springer, Heidelberg, Germany, Apr. 2008, pp. 415–432. DOI: 10.1007/978-3-540-78967-3_24 (cit. on pp. 95, 96).

[Had00] Satoshi Hada. "Zero-Knowledge and Code Obfuscation." In: *Advances in Cryptology – ASIACRYPT 2000*. Ed. by Tatsuaki Okamoto. Vol. 1976. Lecture Notes in Computer Science. Kyoto, Japan: Springer, Heidelberg, Germany, Dec. 2000, pp. 443–457. DOI: 10.1007/3-540-44448-3_34 (cit. on pp. 2, 25, 27, 41).

[HT98] Satoshi Hada and Toshiaki Tanaka. "On the Existence of 3-Round Zero-Knowledge Protocols." In: *Advances in Cryptology – CRYPTO'98*. Ed. by Hugo Krawczyk. Vol. 1462. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 1998, pp. 408–423. DOI: 10.1007/BFb0055744 (cit. on pp. 32, 87).

[HILL99] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. "A Pseudorandom Generator from any One-way Function." In: *SIAM J. Comput.* 28.4 (1999), pp. 1364–1396. DOI: 10.1137/S0097539793244708. URL: https://doi.org/10.1137/S0097539793244708 (cit. on pp. 61, 103, 156, 157, 162, 189, 190, 232).

[HP14] Carmit Hazay and Arpita Patra. "One-Sided Adaptively Secure Two-Party Computation." In: *TCC 2014: 11th Theory of Cryptography Conference*. Ed. by Yehuda Lindell. Vol. 8349. Lecture Notes in Computer Science. San Diego, CA, USA: Springer, Heidelberg, Germany, Feb. 2014, pp. 368–393. DOI: 10.1007/978-3-642-54242-8_16 (cit. on p. 16).

[HLOV11] Brett Hemenway, Benoît Libert, Rafail Ostrovsky, and Damien Vergnaud. "Lossy Encryption: Constructions from General Assumptions and Efficient Selective Opening Chosen Ciphertext Security." In: *Advances in Cryptology – ASIACRYPT 2011*. Ed. by Dong Hoon Lee and Xiaoyun Wang. Vol. 7073. Lecture Notes in Computer Science. Seoul, South Korea: Springer, Heidelberg, Germany, Dec. 2011, pp. 70–88. DOI: 10.1007/978-3-642-25385-0_4 (cit. on p. 97).

[HJK+16] Dennis Hofheinz, Tibor Jager, Dakshita Khurana, Amit Sahai, Brent Waters, and Mark Zhandry. "How to Generate and Use Universal Samplers." In: *Advances in Cryptology – ASIACRYPT 2016, Part II*. Ed. by Jung Hee Cheon and Tsuyoshi Takagi. Vol. 10032. Lecture Notes in Computer Science. Hanoi, Vietnam: Springer, Heidelberg, Germany, Dec. 2016, pp. 715–744. DOI: 10.1007/978-3-662-53890-6_24 (cit. on pp. 150, 161, 171).

[HK08] Dennis Hofheinz and Eike Kiltz. "Programmable Hash Functions and Their Applications." In: *Advances in Cryptology – CRYPTO 2008*. Ed. by David Wagner. Vol. 5157. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 2008, pp. 21–38. DOI: 10.1007/978-3-540-85174-5_2 (cit. on p. 44).

[HK12] Dennis Hofheinz and Eike Kiltz. "Programmable Hash Functions and Their Applications." In: *Journal of Cryptology* 25.3 (July 2012), pp. 484–527. DOI: 10.1007/s00145-011-9102-5 (cit. on pp. 10, 90).

[HMS07] Dennis Hofheinz, John Malone-Lee, and Martijn Stam. "Obfuscation for Cryptographic Purposes." In: *TCC 2007: 4th Theory of Cryptography Conference*. Ed. by Salil P. Vadhan. Vol. 4392. Lecture Notes in Computer Science. Amsterdam, The Netherlands: Springer, Heidelberg, Germany, Feb. 2007, pp. 214–232. DOI: 10.1007/978-3-540-70936-7_12 (cit. on p. 41).

[HRW16] Dennis Hofheinz, Vanishree Rao, and Daniel Wichs. "Standard Security Does Not Imply Indistinguishability Under Selective Opening." In: *TCC 2016-B: 14th Theory of Cryptography Conference, Part II*. Ed. by Martin Hirt and Adam D. Smith. Vol. 9986. Lecture Notes in Computer Science. Beijing, China: Springer, Heidelberg, Germany, Oct. 2016, pp. 121–145. DOI: 10.1007/978-3-662-53644-5_5 (cit. on p. 41).

[HU19] Dennis Hofheinz and Bogdan Ursu. "Dual-Mode NIZKs from Obfuscation." In: *Advances in Cryptology – ASIACRYPT 2019, Part I*. Ed. by Steven D. Galbraith and Shiho Moriai. Vol. 11921. Lecture Notes in Computer Science. Kobe, Japan: Springer,

Heidelberg, Germany, Dec. 2019, pp. 311–341. DOI: 10.1007/978-3-030-34578-5_12 (cit. on pp. 89, 96).

[HRsV07] Susan Hohenberger, Guy N. Rothblum, abhi shelat, and Vinod Vaikuntanathan. "Securely Obfuscating Re-encryption." In: *TCC 2007: 4th Theory of Cryptography Conference*. Ed. by Salil P. Vadhan. Vol. 4392. Lecture Notes in Computer Science. Amsterdam, The Netherlands: Springer, Heidelberg, Germany, Feb. 2007, pp. 233–252. DOI: 10.1007/978-3-540-70936-7_13 (cit. on p. 41).

[HSW13] Susan Hohenberger, Amit Sahai, and Brent Waters. "Full Domain Hash from (Leveled) Multilinear Maps and Identity-Based Aggregate Signatures." In: *Advances in Cryptology – CRYPTO 2013, Part I*. Ed. by Ran Canetti and Juan A. Garay. Vol. 8042. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 2013, pp. 494–512. DOI: 10.1007/978-3-642-40041-4_27 (cit. on pp. 10, 90).

[HSW14] Susan Hohenberger, Amit Sahai, and Brent Waters. "Replacing a Random Oracle: Full Domain Hash from Indistinguishability Obfuscation." In: *Advances in Cryptology – EUROCRYPT 2014*. Ed. by Phong Q. Nguyen and Elisabeth Oswald. Vol. 8441. Lecture Notes in Computer Science. Copenhagen, Denmark: Springer, Heidelberg, Germany, May 2014, pp. 201–220. DOI: 10.1007/978-3-642-55220-5_12 (cit. on pp. 10, 44, 90).

[HW09] Susan Hohenberger and Brent Waters. "Short and Stateless Signatures from the RSA Assumption." In: *Advances in Cryptology – CRYPTO 2009*. Ed. by Shai Halevi. Vol. 5677. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 2009, pp. 654–670. DOI: 10.1007/978-3-642-03356-8_38 (cit. on p. 44).

[HL18] Justin Holmgren and Alex Lombardi. "Cryptographic Hashing from Strong One-Way Functions (Or: One-Way Product Functions and Their Applications)." In: *59th Annual Symposium on Foundations of Computer Science*. Ed. by Mikkel Thorup. Paris, France: IEEE Computer Society Press, Oct. 2018, pp. 850–858. DOI: 10.1109/FOCS.2018.00085 (cit. on p. 3).

[Hop05] Nicholas Hopper. "On Steganographic Chosen Covertext Security." In: *ICALP 2005: 32nd International Colloquium on Automata, Languages and Programming*. Ed. by Luís Caires, Giuseppe F. Italiano, Luís Monteiro, Catuscia Palamidessi, and Moti Yung. Vol. 3580. Lecture Notes in Computer Science. Lisbon, Portugal: Springer, Heidelberg, Germany, July 2005, pp. 311–323. DOI: 10.1007/11523468_26 (cit. on pp. 232, 233).

[HPRV19] Thibaut Horel, Sunoo Park, Silas Richelson, and Vinod Vaikuntanathan. "How to Subvert Backdoored Encryption: Security Against Adversaries that Decrypt All Ciphertexts." In: *ITCS 2019: 10th Innovations in Theoretical Computer Science Conference*. Ed. by Avrim Blum. Vol. 124. San Diego, CA, USA: LIPIcs, Jan. 2019, 42:1–42:20. DOI: 10.4230/LIPIcs.ITCS.2019.42 (cit. on pp. 13, 147).

[HLR07] Chun-Yuan Hsiao, Chi-Jen Lu, and Leonid Reyzin. "Conditional Computational Entropy, or Toward Separating Pseudoentropy from Compressibility." In: *Advances in Cryptology – EUROCRYPT 2007*. Ed. by Moni Naor. Vol. 4515. Lecture Notes in Computer Science. Barcelona, Spain: Springer, Heidelberg, Germany, May 2007, pp. 169–186. DOI: 10.1007/978-3-540-72540-4_10 (cit. on pp. 12, 20, 147, 153, 157, 163, 189, 190, 199, 200).

[HW15] Pavel Hubacek and Daniel Wichs. "On the Communication Complexity of Secure Function Evaluation with Long Output." In: *ITCS 2015: 6th Conference on Innovations in Theoretical Computer Science*. Ed. by Tim Roughgarden. Rehovot, Israel: Association for Computing Machinery, Jan. 2015, pp. 163–172. DOI: 10.1145/2688073.2688105 (cit. on p. 3).

[Huf52] David A Huffman. "A method for the construction of minimum-redundancy codes." In: *Proceedings of the IRE* 40.9 (1952), pp. 1098–1101 (cit. on p. 12).

[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*. IEEE Computer Society, 1995, pp. 134–147. DOI: 10.1109/SCT.1995.514853. URL: https://doi.org/10.1109/SCT.1995.514853 (cit. on p. 156).

[IL90] Russell Impagliazzo and Leonid A. Levin. "No Better Ways to Generate Hard NP Instances than Picking Uniformly at Random." In: *31st Annual Symposium on Foundations of Computer Science*. St. Louis, MO, USA: IEEE Computer Society Press, Oct. 1990, pp. 812–821. DOI: 10.1109/FSCS.1990.89604 (cit. on p. 21).

[ILL89] Russell Impagliazzo, Leonid A. Levin, and Michael Luby. "Pseudo-random Generation from one-way functions (Extended Abstracts)." In: *21st Annual ACM Symposium on Theory of Computing*. Seattle, WA, USA: ACM Press, May 1989, pp. 12–24. DOI: 10.1145/73007.73009 (cit. on p. 20).

[IR89] Russell Impagliazzo and Steven Rudich. "Limits on the Provable Consequences of One-Way Permutations." In: *21st Annual ACM Symposium on Theory of Computing*. Seattle, WA, USA: ACM Press, May 1989, pp. 44–61. DOI: 10.1145/73007.73012 (cit. on p. 33).

[IKOS10] Yuval Ishai, Abishek Kumarasubramanian, Claudio Orlandi, and Amit Sahai. "On Invertible Sampling and Adaptive Security." In: *Advances in Cryptology – ASIACRYPT 2010*. Ed. by Masayuki Abe. Vol. 6477. Lecture Notes in Computer Science. Singapore: Springer, Heidelberg, Germany, Dec. 2010, pp. 466–482. DOI: 10.1007/978-3-642-17373-8_27 (cit. on pp. 16, 17, 151–154, 158, 159, 177, 201, 204, 205, 212, 221, 224, 226).

[IPS15] Yuval Ishai, Omkant Pandey, and Amit Sahai. "Public-Coin Differing-Inputs Obfuscation and Its Applications." In: *TCC 2015: 12th Theory of Cryptography Conference, Part II*. Ed. by Yevgeniy Dodis and Jesper Buus Nielsen. Vol. 9015. Lecture Notes in Computer Science. Warsaw, Poland: Springer, Heidelberg, Germany, Mar. 2015, pp. 668–697. DOI: 10.1007/978-3-662-46497-7_26 (cit. on p. 41).

[IPS08] Yuval Ishai, Manoj Prabhakaran, and Amit Sahai. "Founding Cryptography on Oblivious Transfer - Efficiently." In: *Advances in Cryptology – CRYPTO 2008*. Ed. by David Wagner. Vol. 5157. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 2008, pp. 572–591. DOI: 10.1007/978-3-540-85174-5_32 (cit. on pp. 16, 224).

[JRT16] Joseph Jaeger, Thomas Ristenpart, and Qiang Tang. "Honey Encryption Beyond Message Recovery Security." In: *Advances in Cryptology – EUROCRYPT 2016, Part I*. Ed. by Marc Fischlin and Jean-Sébastien Coron. Vol. 9665. Lecture Notes in Computer Science. Vienna, Austria: Springer, Heidelberg, Germany, May 2016, pp. 758–788. DOI: 10.1007/978-3-662-49890-3_29 (cit. on p. 230).

[JLMS19] Aayush Jain, Huijia Lin, Christian Matt, and Amit Sahai. "How to Leverage Hardness of Constant-Degree Expanding Polynomials overa $\mathbb{R}$ to build $i\mathcal{O}$." In: *Advances in Cryptology – EUROCRYPT 2019, Part I*. Ed. by Yuval Ishai and Vincent Rijmen. Vol. 11476. Lecture Notes in Computer Science. Darmstadt, Germany: Springer, Heidelberg, Germany, May 2019, pp. 251–281. DOI: 10.1007/978-3-030-17653-2_9 (cit. on pp. 2, 41).

[JLS20] Aayush Jain, Huijia Lin, and Amit Sahai. *Indistinguishability Obfuscation from Well-Founded Assumptions*. Cryptology ePrint Archive, Report 2020/1003. https://eprint.iacr.org/2020/1003. 2020 (cit. on pp. 2, 5, 27, 41).

[Jak98] Markus Jakobsson. "A Practical Mix." In: *Advances in Cryptology – EUROCRYPT'98*. Ed. by Kaisa Nyberg. Vol. 1403. Lecture Notes in Computer Science. Espoo, Finland: Springer, Heidelberg, Germany, May 1998, pp. 448–461. DOI: 10.1007/BFb0054145 (cit. on pp. 89, 129).

[JR14] Ari Juels and Thomas Ristenpart. "Honey Encryption: Security Beyond the Brute-Force Bound." In: *Advances in Cryptology – EUROCRYPT 2014*. Ed. by Phong Q. Nguyen and Elisabeth Oswald. Vol. 8441. Lecture Notes in Computer Science. Copenhagen, Denmark: Springer, Heidelberg, Germany, May 2014, pp. 293–310. DOI: 10.1007/978-3-642-55220-5_17 (cit. on pp. 13, 147, 153, 156, 158, 221, 229–231).

[KLMR18] Yuan Kang, Chengyu Lin, Tal Malkin, and Mariana Raykova. "Obfuscation from Polynomial Hardness: Beyond Decomposable Obfuscation." In: *SCN 18: 11th International Conference on Security in Communication Networks*. Ed. by Dario Catalano and Roberto De Prisco. Vol. 11035. Lecture Notes in Computer Science. Amalfi, Italy: Springer, Heidelberg, Germany, Sept. 2018,

pp. 407–424. DOI: 10.1007/978-3-319-98113-0_22 (cit. on p. 5).

[KLRX20] Julia Kastner, Julian Loss, Michael Rosenberg, and Jiayu Xu. *On Pairing-Free Blind Signature Schemes in the Algebraic Group Model*. Cryptology ePrint Archive, Report 2020/1071. https://eprint.iacr.org/2020/1071. 2020 (cit. on p. 245).

[KP19] Julia Kastner and Jiaxin Pan. *Towards Instantiating the Algebraic Group Model*. Cryptology ePrint Archive, Report 2019/1018. https://eprint.iacr.org/2019/1018. 2019 (cit. on pp. 9, 88, 92, 94, 109, 110).

[Kat08] Jonathan Katz. "Which Languages Have 4-Round Zero-Knowledge Proofs?" In: *TCC 2008: 5th Theory of Cryptography Conference*. Ed. by Ran Canetti. Vol. 4948. Lecture Notes in Computer Science. San Francisco, CA, USA: Springer, Heidelberg, Germany, Mar. 2008, pp. 73–88. DOI: 10.1007/978-3-540-78524-8_5 (cit. on p. 32).

[KO04] Jonathan Katz and Rafail Ostrovsky. "Round-Optimal Secure Two-Party Computation." In: *Advances in Cryptology – CRYPTO 2004*. Ed. by Matthew Franklin. Vol. 3152. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 2004, pp. 335–354. DOI: 10.1007/978-3-540-28628-8_21 (cit. on p. 16).

[KPTZ13] Aggelos Kiayias, Stavros Papadopoulos, Nikos Triandopoulos, and Thomas Zacharias. "Delegatable pseudorandom functions and applications." In: *ACM CCS 2013: 20th Conference on Computer and Communications Security*. Ed. by Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung. Berlin, Germany: ACM Press, Nov. 2013, pp. 669–684. DOI: 10.1145/2508859.2516668 (cit. on pp. 23, 24).

[Kil88] Joe Kilian. "Founding Cryptography on Oblivious Transfer." In: *20th Annual ACM Symposium on Theory of Computing*. Chicago, IL, USA: ACM Press, May 1988, pp. 20–31. DOI: 10.1145/62212.62215 (cit. on p. 224).

[Kol68] Andrei Nikolaevich Kolmogorov. "Three approaches to the quantitative definition of information." In: *International journal of computer mathematics* 2.1-4 (1968), pp. 157–168 (cit. on pp. 13, 147).

[KNY14] Ilan Komargodski, Moni Naor, and Eylon Yogev. "Secret-Sharing for NP." In: *Advances in Cryptology – ASIACRYPT 2014, Part II*. Ed. by Palash Sarkar and Tetsu Iwata. Vol. 8874. Lecture Notes in Computer Science. Kaoshiung, Taiwan, R.O.C.: Springer, Heidelberg, Germany, Dec. 2014, pp. 254–273. DOI: 10.1007/978-3-662-45608-8_14 (cit. on p. 3).

[KLW15] Venkata Koppula, Allison Bishop Lewko, and Brent Waters. "Indistinguishability Obfuscation for Turing Machines with Unbounded Memory." In: *47th Annual ACM Symposium on Theory of Computing*. Ed. by Rocco A. Servedio and Ronitt Rubinfeld. Portland, OR, USA: ACM Press, June 2015, pp. 419–428. DOI: 10.1145/2746539.2746614 (cit. on p. 3).

[LMs05] Matt Lepinski, Silvio Micali, and abhi shelat. "Fair-Zero Knowledge." In: *TCC 2005: 2nd Theory of Cryptography Conference*. Ed. by Joe Kilian. Vol. 3378. Lecture Notes in Computer Science. Cambridge, MA, USA: Springer, Heidelberg, Germany, Feb. 2005, pp. 245–263. DOI: 10.1007/978-3-540-30576-7_14 (cit. on pp. 153, 163, 199).

[Lev86] Leonid A. Levin. "Average Case Complete Problems." In: *SIAM J. Comput.* 15.1 (1986), pp. 285–286. DOI: 10.1137/0215020. URL: https://doi.org/10.1137/0215020 (cit. on pp. 1, 21).

[LM16] Baiyu Li and Daniele Micciancio. "Compactness vs Collusion Resistance in Functional Encryption." In: *TCC 2016-B: 14th Theory of Cryptography Conference, Part II*. Ed. by Martin Hirt and Adam D. Smith. Vol. 9986. Lecture Notes in Computer Science. Beijing, China: Springer, Heidelberg, Germany, Oct. 2016, pp. 443–468. DOI: 10.1007/978-3-662-53644-5_17 (cit. on p. 42).

[LV90] Ming Li and Paul M. B. Vitányi. "Handbook of Theoretical Computer Science (Vol. A)." In: ed. by Jan van Leeuwen. Cambridge, MA, USA: MIT Press, 1990. Chap. Kolmogorov Complexity and Its Applications, pp. 187–254. ISBN: 0-444-88071-2. URL: http://dl.acm.org/citation.cfm?id=114872.114876 (cit. on pp. 13, 147).

[LV19] Ming Li and Paul M. B. Vitányi. *An Introduction to Kolmogorov Complexity and Its Applications, 4th Edition*. Texts in Computer Science. Springer, 2019. ISBN: 978-3-030-11297-4. DOI: 10.1007/978-3-030-11298-1 (cit. on pp. 13, 147).

[Lin16] Huijia Lin. "Indistinguishability Obfuscation from Constant-Degree Graded Encoding Schemes." In: *Advances in Cryptology – EUROCRYPT 2016, Part I*. Ed. by Marc Fischlin and Jean-Sébastien Coron. Vol. 9665. Lecture Notes in Computer Science. Vienna, Austria: Springer, Heidelberg, Germany, May 2016, pp. 28–57. DOI: 10.1007/978-3-662-49890-3_2 (cit. on p. 2).

[Lin17] Huijia Lin. "Indistinguishability Obfuscation from SXDH on 5-Linear Maps and Locality-5 PRGs." In: *Advances in Cryptology – CRYPTO 2017, Part I*. Ed. by Jonathan Katz and Hovav Shacham. Vol. 10401. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 2017, pp. 599–629. DOI: 10.1007/978-3-319-63688-7_20 (cit. on pp. 2, 41).

[LT17] Huijia Lin and Stefano Tessaro. "Indistinguishability Obfuscation from Trilinear Maps and Block-Wise Local PRGs." In: *Advances in Cryptology – CRYPTO 2017, Part I*. Ed. by Jonathan Katz and Hovav Shacham. Vol. 10401. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 2017, pp. 630–660. DOI: 10.1007/978-3-319-63688-7_21 (cit. on pp. 2, 41).

[LV16] Huijia Lin and Vinod Vaikuntanathan. "Indistinguishability Obfuscation from DDH-Like Assumptions on Constant-Degree Graded Encodings." In: *57th Annual Symposium on Foundations of Computer Science*. Ed. by Irit Dinur. New Brunswick, NJ, USA: IEEE Computer Society Press, Oct. 2016, pp. 11–20. DOI: 10.1109/FOCS.2016.11 (cit. on p. 2).

[Lin09] Andrew Y. Lindell. "Adaptively Secure Two-Party Computation with Erasures." In: *Topics in Cryptology – CT-RSA 2009*. Ed. by Marc Fischlin. Vol. 5473. Lecture Notes in Computer Science. San Francisco, CA, USA: Springer, Heidelberg, Germany, Apr. 2009, pp. 117–132. DOI: 10.1007/978-3-642-00862-7_8 (cit. on p. 222).

[LZ17] Qipeng Liu and Mark Zhandry. "Decomposable Obfuscation: A Framework for Building Applications of Obfuscation from Polynomial Hardness." In: *TCC 2017: 15th Theory of Cryptography Conference, Part I*. Ed. by Yael Kalai and Leonid Reyzin. Vol. 10677. Lecture Notes in Computer Science. Baltimore, MD, USA: Springer, Heidelberg, Germany, Nov. 2017, pp. 138–169. DOI: 10.1007/978-3-319-70500-2_6 (cit. on pp. 5, 42, 44).

[LPS04] Ben Lynn, Manoj Prabhakaran, and Amit Sahai. "Positive Results and Techniques for Obfuscation." In: *Advances in Cryptology – EUROCRYPT 2004*. Ed. by Christian Cachin and Jan Camenisch. Vol. 3027. Lecture Notes in Computer Science. Interlaken, Switzerland: Springer, Heidelberg, Germany, May 2004, pp. 20–39. DOI: 10.1007/978-3-540-24676-3_2 (cit. on p. 41).

[Mau05] Ueli M. Maurer. "Abstract Models of Computation in Cryptography (Invited Paper)." In: *10th IMA International Conference on Cryptography and Coding*. Ed. by Nigel P. Smart. Vol. 3796. Lecture Notes in Computer Science. Cirencester, UK: Springer, Heidelberg, Germany, Dec. 2005, pp. 1–12 (cit. on pp. 8, 87, 93).

[MW98] Ueli M. Maurer and Stefan Wolf. "Lower Bounds on Generic Algorithms in Groups." In: *Advances in Cryptology – EUROCRYPT'98*. Ed. by Kaisa Nyberg. Vol. 1403. Lecture Notes in Computer Science. Espoo, Finland: Springer, Heidelberg, Germany, May 1998, pp. 72–84. DOI: 10.1007/BFb0054118 (cit. on p. 87).

[MG02] Daniele Micciancio and Shafi Goldwasser. *Complexity of lattice problems - a cryptograhic perspective*. Vol. 671. The Kluwer international series in engineering and computer science. Springer, 2002. ISBN: 978-0-7923-7688-0 (cit. on p. 106).

[MSZ16] Eric Miles, Amit Sahai, and Mark Zhandry. "Annihilation Attacks for Multilinear Maps: Cryptanalysis of Indistinguishability Obfuscation over GGH13." In: *Advances in Cryptology – CRYPTO 2016, Part II*. Ed. by Matthew Robshaw and Jonathan Katz. Vol. 9815. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 2016,

pp. 629–658. DOI: 10.1007/978-3-662-53008-5_22 (cit. on p. 2).

[Nao03] Moni Naor. "On Cryptographic Assumptions and Challenges (Invited Talk)." In: *Advances in Cryptology – CRYPTO 2003*. Ed. by Dan Boneh. Vol. 2729. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 2003, pp. 96–109. DOI: 10.1007/978-3-540-45146-4_6 (cit. on pp. 10, 22, 27, 32, 88, 94).

[Nec94] V. I. Nechaev. "Complexity of a Determinate Algorithm for the Discrete Logarithm." In: *Mathematical Notes* 55.2 (1994), pp. 165–172 (cit. on p. 87).

[NZ96] Noam Nisan and David Zuckerman. "Randomness is Linear in Space." In: *J. Comput. Syst. Sci.* 52.1 (1996), pp. 43–52. DOI: 10.1006/jcss.1996.0004. URL: https://doi.org/10.1006/jcss.1996.0004 (cit. on p. 20).

[OPWW15] Tatsuaki Okamoto, Krzysztof Pietrzak, Brent Waters, and Daniel Wichs. "New Realizations of Somewhere Statistically Binding Hashing and Positional Accumulators." In: *Advances in Cryptology – ASIACRYPT 2015, Part I*. Ed. by Tetsu Iwata and Jung Hee Cheon. Vol. 9452. Lecture Notes in Computer Science. Auckland, New Zealand: Springer, Heidelberg, Germany, Nov. 2015, pp. 121–145. DOI: 10.1007/978-3-662-48797-6_6 (cit. on p. 3).

[OW93] Rafail Ostrovsky and Avi Wigderson. "One-Way Functions are Essential for Non-Trivial Zero-Knowledge." In: *Second Israel Symposium on Theory of Computing Systems, ISTCS 1993, Natanya, Israel, June 7-9, 1993, Proceedings*. IEEE Computer Society, 1993, pp. 3–17. DOI: 10.1109/ISTCS.1993.253489. URL: https://doi.org/10.1109/ISTCS.1993.253489 (cit. on p. 35).

[Pai99] Pascal Paillier. "Public-Key Cryptosystems Based on Composite Degree Residuosity Classes." In: *Advances in Cryptology – EUROCRYPT'99*. Ed. by Jacques Stern. Vol. 1592. Lecture Notes in Computer Science. Prague, Czech Republic: Springer, Heidelberg, Germany, May 1999, pp. 223–238. DOI: 10.1007/3-540-48910-X_16 (cit. on p. 97).

[PV05] Pascal Paillier and Damien Vergnaud. "Discrete-Log-Based Signatures May Not Be Equivalent to Discrete Log." In: *Advances in Cryptology – ASIACRYPT 2005*. Ed. by Bimal K. Roy. Vol. 3788. Lecture Notes in Computer Science. Chennai, India: Springer, Heidelberg, Germany, Dec. 2005, pp. 1–20. DOI: 10.1007/11593447_1 (cit. on p. 87).

[Pas77] Richard C. Pasco. "Source coding algorithms for fast data compression (Ph.D. Thesis abstr.)" In: *IEEE Trans. Inf. Theory* 23.4 (1977), p. 548. DOI: 10.1109/TIT.1977.1055739. URL: https://doi.org/10.1109/TIT.1977.1055739 (cit. on p. 12).

[Pas03] Rafael Pass. "Simulation in Quasi-Polynomial Time, and Its Application to Protocol Composition." In: *Advances in Cryptology – EUROCRYPT 2003*. Ed. by Eli Biham. Vol. 2656. Lecture Notes in Computer Science. Warsaw, Poland: Springer, Heidelberg, Germany, May 2003, pp. 160–176. DOI: 10.1007/3-540-39200-9_10 (cit. on p. 241).

[PST14] Rafael Pass, Karn Seth, and Sidharth Telang. "Indistinguishability Obfuscation from Semantically-Secure Multilinear Encodings." In: *Advances in Cryptology – CRYPTO 2014, Part I*. Ed. by Juan A. Garay and Rosario Gennaro. Vol. 8616. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 2014, pp. 500–517. DOI: 10.1007/978-3-662-44371-2_28 (cit. on pp. 41, 43).

[Ps16] Rafael Pass and abhi shelat. "Impossibility of VBB Obfuscation with Ideal Constant-Degree Graded Encodings." In: *TCC 2016-A: 13th Theory of Cryptography Conference, Part I*. Ed. by Eyal Kushilevitz and Tal Malkin. Vol. 9562. Lecture Notes in Computer Science. Tel Aviv, Israel: Springer, Heidelberg, Germany, Jan. 2016, pp. 3–17. DOI: 10.1007/978-3-662-49096-9_1 (cit. on p. 42).

[PS19] Chris Peikert and Sina Shiehian. "Noninteractive Zero Knowledge for NP from (Plain) Learning with Errors." In: *Advances in Cryptology – CRYPTO 2019, Part I*. Ed. by Alexandra Boldyreva and Daniele Micciancio. Vol. 11692. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 2019, pp. 89–114. DOI: 10.1007/978-3-030-26948-7_4 (cit. on pp. 3, 36, 96).

[PR07] Manoj Prabhakaran and Mike Rosulek. "Rerandomizable RCCA Encryption." In: *Advances in Cryptology – CRYPTO 2007*. Ed. by Alfred Menezes. Vol. 4622. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 2007, pp. 517–534. DOI: 10.1007/978-3-540-74143-5_29 (cit. on p. 97).

[RS92] Charles Rackoff and Daniel R. Simon. "Non-Interactive Zero-Knowledge Proof of Knowledge and Chosen Ciphertext Attack." In: *Advances in Cryptology – CRYPTO'91*. Ed. by Joan Feigenbaum. Vol. 576. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 1992, pp. 433–444. DOI: 10.1007/3-540-46766-1_35 (cit. on p. 129).

[RR99] Ran Raz and Omer Reingold. "On Recycling the Randomness of States in Space Bounded Computation." In: *31st Annual ACM Symposium on Theory of Computing*. Atlanta, GA, USA: ACM Press, May 1999, pp. 159–168. DOI: 10.1145/301250.301294 (cit. on pp. 13, 147).

[Rey11] Leonid Reyzin. "Some Notions of Entropy for Cryptography - (Invited Talk)." In: *ICITS 11: 5th International Conference on Information Theoretic Security*. Ed. by Serge Fehr. Vol. 6673. Lecture Notes in Computer Science. Amsterdam, The Netherlands:

Springer, Heidelberg, Germany, May 2011, pp. 138–142. DOI: 10.1007/978-3-642-20728-0_13 (cit. on p. 189).

[Ris76] Jorma Rissanen. "Generalized Kraft Inequality and Arithmetic Coding." In: *IBM J. Res. Dev.* 20.3 (1976), pp. 198–203. DOI: 10.1147/rd.203.0198. URL: https://doi.org/10.1147/rd.203.0198 (cit. on p. 12).

[RJ79] Jorma Rissanen and Glen G. Langdon Jr. "Arithmetic Coding." In: *IBM J. Res. Dev.* 23.2 (1979), pp. 149–162. DOI: 10.1147/rd.232.0149. URL: https://doi.org/10.1147/rd.232.0149 (cit. on p. 12).

[RAD78] Ronald L. Rivest, Leonard M. Adleman, and Michael L. Dertouzos. "On data banks and privacy homomorphisms." In: *Foundations of secure computation* 4.11 (1978), pp. 169–180 (cit. on p. 34).

[RSA78] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems." In: *Commun. ACM* 21.2 (1978), pp. 120–126. DOI: 10.1145/359340.359342. URL: http://doi.acm.org/10.1145/359340.359342 (cit. on p. 2).

[RS20] Lior Rotem and Gil Segev. "Algebraic Distinguishers: From Discrete Logarithms to Decisional Uber Assumptions." In: (2020) (cit. on p. 245).

[SW14] Amit Sahai and Brent Waters. "How to use indistinguishability obfuscation: deniable encryption, and more." In: *46th Annual ACM Symposium on Theory of Computing*. Ed. by David B. Shmoys. New York, NY, USA: ACM Press, May 2014, pp. 475–484. DOI: 10.1145/2591796.2591825 (cit. on pp. 3, 24, 41, 43, 103, 104, 152, 153, 155, 189, 200, 212–215, 233).

[San87] Miklos Santha. "On Using Deterministic Functions to Reduce Randomness in Probabilistic Algorithms." In: *Inf. Comput.* 74.3 (1987), pp. 241–249. DOI: 10.1016/0890-5401(87)90023-X. URL: https://doi.org/10.1016/0890-5401(87)90023-X (cit. on p. 20).

[Sch72] J. Pieter M. Schalkwijk. "An algorithm for source coding." In: *IEEE Trans. Inf. Theory* 18.3 (1972), pp. 395–399. DOI: 10.1109/TIT.1972.1054832. URL: https://doi.org/10.1109/TIT.1972.1054832 (cit. on p. 12).

[Sch91] Claus-Peter Schnorr. "Efficient Signature Generation by Smart Cards." In: *Journal of Cryptology* 4.3 (Jan. 1991), pp. 161–174. DOI: 10.1007/BF00196725 (cit. on p. 89).

[SJ00] Claus-Peter Schnorr and Markus Jakobsson. "Security of Signed ElGamal Encryption." In: *Advances in Cryptology – ASIACRYPT 2000*. Ed. by Tatsuaki Okamoto. Vol. 1976. Lecture Notes in Computer Science. Kyoto, Japan: Springer, Heidelberg, Germany, Dec. 2000, pp. 73–89. DOI: 10.1007/3-540-44448-3_7 (cit. on p. 129).

[Sha48] Claude E. Shannon. "A mathematical theory of communication." In: *Bell Syst. Tech. J.* 27.3 (1948), pp. 379–423. DOI: 10.1002/j.1538-7305.1948.tb01338.x. URL: https://doi.org/10.1002/j.1538-7305.1948.tb01338.x (cit. on pp. 12, 162, 190).

[TUZ01] Amnon Ta-Shma, Christopher Umans, and David Zuckerman. "Loss-less condensers, unbalanced expanders, and extractors." In: *33rd Annual ACM Symposium on Theory of Computing*. Crete, Greece: ACM Press, July 2001, pp. 143–152. DOI: 10.1145/380752.380790 (cit. on pp. 13, 147).

[Sho97] Victor Shoup. "Lower Bounds for Discrete Logarithms and Related Problems." In: *Advances in Cryptology – EUROCRYPT'97*. Ed. by Walter Fumy. Vol. 1233. Lecture Notes in Computer Science. Konstanz, Germany: Springer, Heidelberg, Germany, May 1997, pp. 256–266. DOI: 10.1007/3-540-69053-0_18 (cit. on pp. 8, 10, 87, 93).

[Sho04] Victor Shoup. *Sequences of games: a tool for taming complexity in security proofs*. Cryptology ePrint Archive, Report 2004/332. http://eprint.iacr.org/2004/332. 2004 (cit. on pp. 172, 185, 186).

[Sip88] Michael Sipser. "Expanders, Randomness, or Time versus Space." In: *J. Comput. Syst. Sci.* 36.3 (1988), pp. 379–383. DOI: 10.1016/0022-0000(88)90035-9. URL: https://doi.org/10.1016/0022-0000(88)90035-9 (cit. on p. 20).

[Sol64] Ray J. Solomonoff. "A Formal Theory of Inductive Inference. Part I." In: *Inf. Control.* 7.1 (1964), pp. 1–22. DOI: 10.1016/S0019-9958(64)90223-2. URL: https://doi.org/10.1016/S0019-9958(64)90223-2 (cit. on pp. 13, 147).

[TV00] Luca Trevisan and Salil P. Vadhan. "Extracting Randomness from Samplable Distributions." In: *41st Annual Symposium on Foundations of Computer Science*. Redondo Beach, CA, USA: IEEE Computer Society Press, Nov. 2000, pp. 32–42. DOI: 10.1109/SFCS.2000.892063 (cit. on pp. 13, 147).

[TVZ05] Luca Trevisan, Salil P. Vadhan, and David Zuckerman. "Compression of Samplable Sources." In: *Computational Complexity* 14.3 (2005), pp. 186–227. DOI: 10.1007/s00037-005-0198-6. URL: https://doi.org/10.1007/s00037-005-0198-6 (cit. on pp. 12, 13, 147, 156, 162, 168, 190, 191).

[TY98] Yiannis Tsiounis and Moti Yung. "On the Security of ElGamal Based Encryption." In: *PKC'98: 1st International Workshop on Theory and Practice in Public Key Cryptography*. Ed. by Hideki Imai and Yuliang Zheng. Vol. 1431. Lecture Notes in Computer Science. Pacifico Yokohama, Japan: Springer, Heidelberg, Germany, Feb. 1998, pp. 117–134. DOI: 10.1007/BFb0054019 (cit. on pp. 89, 129).

[Tur50] AM Turing. "Mind." In: *Mind* 59.236 (1950), pp. 433–460 (cit. on p. vii).

[Val76] Leslie G. Valiant. "Relative Complexity of Checking and Evaluating." In: *Inf. Process. Lett.* 5.1 (1976), pp. 20–23. DOI: 10.1016/0020-0190(76)90097-1. URL: https://doi.org/10.1016/0020-0190(76)90097-1 (cit. on p. 94).

[vH04] Luis von Ahn and Nicholas J. Hopper. "Public-Key Steganography." In: *Advances in Cryptology – EUROCRYPT 2004*. Ed. by Christian Cachin and Jan Camenisch. Vol. 3027. Lecture Notes in Computer Science. Interlaken, Switzerland: Springer, Heidelberg, Germany, May 2004, pp. 323–341. DOI: 10.1007/978-3-540-24676-3_20 (cit. on pp. 232, 233).

[vHL05] Luis von Ahn, Nicholas J. Hopper, and John Langford. "Covert two-party computation." In: *37th Annual ACM Symposium on Theory of Computing*. Ed. by Harold N. Gabow and Ronald Fagin. Baltimore, MA, USA: ACM Press, May 2005, pp. 513–522. DOI: 10.1145/1060590.1060668 (cit. on pp. 13, 147, 155, 245).

[Wat05] Brent R. Waters. "Efficient Identity-Based Encryption Without Random Oracles." In: *Advances in Cryptology – EUROCRYPT 2005*. Ed. by Ronald Cramer. Vol. 3494. Lecture Notes in Computer Science. Aarhus, Denmark: Springer, Heidelberg, Germany, May 2005, pp. 114–127. DOI: 10.1007/11426639_7 (cit. on p. 44).

[Wee04] Hoeteck Wee. "On Pseudoentropy versus Compressibility." In: *19th Annual IEEE Conference on Computational Complexity (CCC 2004), 21-24 June 2004, Amherst, MA, USA*. IEEE Computer Society, 2004, pp. 29–41. DOI: 10.1109/CCC.2004.1313782. URL: https://doi.org/10.1109/CCC.2004.1313782 (cit. on pp. 12, 147, 157, 162, 190).

[Wee05] Hoeteck Wee. "On obfuscating point functions." In: *37th Annual ACM Symposium on Theory of Computing*. Ed. by Harold N. Gabow and Ronald Fagin. Baltimore, MA, USA: ACM Press, May 2005, pp. 523–532. DOI: 10.1145/1060590.1060669 (cit. on p. 41).

[WW20] Hoeteck Wee and Daniel Wichs. *Candidate Obfuscation via Oblivious LWE Sampling*. Cryptology ePrint Archive, Report 2020/1042. https://eprint.iacr.org/2020/1042. 2020 (cit. on pp. 2, 17, 41, 158, 159, 245).

[WZ17] Daniel Wichs and Giorgos Zirdelis. "Obfuscating Compute-and-Compare Programs under LWE." In: *58th Annual Symposium on Foundations of Computer Science*. Ed. by Chris Umans. Berkeley, CA, USA: IEEE Computer Society Press, Oct. 2017, pp. 600–611. DOI: 10.1109/FOCS.2017.61 (cit. on p. 3).

[WS07] J. Wu and D.R. Stinson. *An Efficient Identification Protocol and the Knowledge-of-Exponent Assumption*. Cryptology ePrint Archive, Report 2007/479. http://eprint.iacr.org/2007/479. 2007 (cit. on pp. 9, 88, 109).

[Yao82] Andrew Chi-Chih Yao. "Theory and Applications of Trapdoor Functions (Extended Abstract)." In: *23rd Annual Symposium on Foundations of Computer Science*. Chicago, Illinois: IEEE Computer Society Press, Nov. 1982, pp. 80–91. DOI: 10.1109/SFCS.1982.45 (cit. on pp. 21, 33, 157, 162, 190).

[Zha16] Mark Zhandry. "The Magic of ELFs." In: *Advances in Cryptology – CRYPTO 2016, Part I*. Ed. by Matthew Robshaw and Jonathan Katz. Vol. 9814. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 2016, pp. 479–508. DOI: 10.1007/978-3-662-53018-4_18 (cit. on pp. 7, 42, 44, 45, 47–49, 153, 238–240).

[Zim15] Joe Zimmerman. "How to Obfuscate Programs Directly." In: *Advances in Cryptology – EUROCRYPT 2015, Part II*. Ed. by Elisabeth Oswald and Marc Fischlin. Vol. 9057. Lecture Notes in Computer Science. Sofia, Bulgaria: Springer, Heidelberg, Germany, Apr. 2015, pp. 439–467. DOI: 10.1007/978-3-662-46803-6_15 (cit. on p. 41).

[ZL77] Jacob Ziv and Abraham Lempel. "A universal algorithm for sequential data compression." In: *IEEE Trans. Inf. Theory* 23.3 (1977), pp. 337–343. DOI: 10.1109/TIT.1977.1055714. URL: https://doi.org/10.1109/TIT.1977.1055714 (cit. on p. 12).

[ZL78] Jacob Ziv and Abraham Lempel. "Compression of individual sequences via variable-rate coding." In: *IEEE Trans. Inf. Theory* 24.5 (1978), pp. 530–536. DOI: 10.1109/TIT.1978.1055934. URL: https://doi.org/10.1109/TIT.1978.1055934 (cit. on p. 12).

COLOPHON

This document was typeset with LaTeX 2ε using the typographical look-and-feel `classicthesis` developed by André Miede and Ivo Pletikosić using Hermann Zapf's *Palatino* and *Euler* type faces (Type 1 PostScript fonts *URW Palladio L* and *FPL* were used). The style was inspired by Robert Bringhurst's seminal book on typography *"The Elements of Typographic Style"*.

Final version as of June 15, 2021.