

**Getting inside the Adversary's Head:
New Directions in
Non-Black-Box Knowledge Extraction**

Thesis submitted for the degree of Doctor of Philosophy
by
Nir Bitansky

This work was carried out under the supervision of
Professor Ran Canetti

Submitted to the Senate of Tel Aviv University
August 2014

© 2014

Copyright by Nir Bitansky

All Rights Reserved

To my family.

Acknowledgements

I am deeply grateful to my advisor, Ran Canetti. The Kolmogorov complexity of my gratitude exceeds by far the few lines that I have here, but I'll make an effort.

Ran had quite the substantial effect on shaping me as a researcher and presenter of ideas. He has always offered his angle and (often strong) opinion on research, presentation, and beyond. Even after some years absorbing these opinions, I still find them to be quite unsimulatable, and often strikingly valuable. At the same time, Ran has never seemed to underestimate the less experienced (and at times stubborn) voice, has always kept an open mind, and was more than willing to debate. I believe I have gained much from this process, and am thankful for the energy that Ran spent within.

Beyond his qualities as an advisor, Ran is just a fun person to be around. Some of my favorite properties of his include: his hospitality, which somehow extends beyond his home, and which goes well with his passion to food, his sense of humor, and his colorful taste in fashion (whenever Ran walked in to the room, I would have an urge to announce "TAKI!" [2]).

A special thanks goes to Omer Paneth. Collaborating with Omer has always been extraordinary, and almost addictive. It's hard to characterize exactly why, but it must have something to do with Omer's way of finding intuition where there seems to be none, his endless patience to listen, explain, and understand, his wild thoughts, unique sense of humor, and the fact that his father invented the internet. Besides that, Omer is a true friend.

Another special thank you goes to Alessandro Chiesa. I feel truly fortunate to have collaborated with Alessandro. It was more than once that I was swept off my feet by his endless energy and optimism. Besides being a great researcher, Ale is also an intriguing person, so serious but yet so full of humor, and the only true ideologist I know.

I would also like to thank the members (or fellow interns) of the crypto group at IBM TJ Watson Research Center: Charanjit, Craig, Daniel, David, Hugo, Mark, Sanjam, Shai, Tal, and Victor, for two unforgettable summers. Special thanks to Daniel and

Victor for tolerating my presence on the way to and from IBM. Another special thank you to Tal, for your embracing hospitality, support, and advice throughout the time we have known each other. You will forever be in my thoughts during lunch.

Since I started my journey in research, I've encountered quite a few tremendous people from which I have learned much and view as role models, especially: Benny Chor, Shai Halevi, Yuval Ishai, Yael Kalai, and Alon Rosen. I am grateful to you all.

Throughout my studies, I was fortunate to collaborate with remarkable researchers (some of which I've already mentioned), which I am thankful for. The results in this thesis reflect some of these fruitful collaborations. In particular, Chapter 3 is based on joint work with Omer, Chapter 4 is based on joint work with Ran, Omer, and Alon, and Chapter 5 is based on joint work with Ran, Ale, and Eran Tromer.

I thank my fellow students Ben, Itai, and Rita at the crypto group at Tel University, and the crypto and security group that hosted me during my visit to Boston University: Adam, Ben, Leo, Robert, Rachel (also one of my favorite collaborators) and (my cousin) Sharon.

Last but not least, I thank Sagit, my better half. I could not have done this without your support. You are the blood in my veins, the fire in my furnace, the piggy in my blanket.

Abstract

Arguing about *the knowledge of the adversary* in the context of a given scheme or protocol is central to modern cryptographic analysis. Typically, such knowledge is captured by our ability to algorithmically extract it from the adversary’s program. Following a long tradition of black-box reductions (or simulators), most knowledge extractors treat the adversary’s program as a black-box, independently of its actual code implementation. However, as several lower-bounds imply, black-box extraction is insufficient for certain desirable cryptographic goals. Quite remarkably, Barak and subsequent extensions show that some of these goals can be achieved based on *non-black-box techniques*.

We develop new techniques for non-black-box knowledge extraction from adversarial code. We demonstrate the power of these techniques in several new cryptographic schemes and protocols achieving features that were previously out of reach. Throughout the thesis, we put special focus on the contrast between code obfuscation and non-black-box extraction, revealing new points on the tradeoff between the two.

Concretely, this thesis includes three main contributions:

Unobfuscatable functions and applications to resettable cryptography. We prove a new impossibility result for a notion of obfuscation *with approximate functionality*, demonstrating the existence of certain *robust unobfuscatable functions*. Relying on the constructed robust unobfuscatable functions, we obtain a new non-black-box extraction technique that yields new zero-knowledge protocols with resilience to various *resetting attacks*. The protocols feature improved round-complexity, and weaker computational assumptions than known before. The non-black-box zero-knowledge simulation in these protocols is substantially different from that of Barak.

On the existence of extractable functions. A function f is extractable if it is possible to algorithmically “extract,” from any adversarial program that outputs a value v in the image of f , a preimage of v . When combined with hardness properties

such as one-wayness, extractable functions are known to achieve cryptographic goals which are out of the reach of black-box techniques, Barak’s non-black-box techniques, as well as the new techniques mentioned above.

So far, extractability has not been explicitly shown. Instead, it has only been considered as a non-standard *knowledge assumption* on certain functions. We make two headways in the study of extractable one-way functions (EOWFs). On the negative side, we show that if there exist indistinguishability obfuscators for a certain class of circuits then there do not exist EOWFs where extraction works for any adversarial program with auxiliary-input of unbounded polynomial length. On the positive side, for adversarial programs with bounded auxiliary-input (and unbounded polynomial running time), we give the first construction of EOWFs with an explicit extraction procedure, based on relatively standard assumptions (e.g., sub-exponential hardness of Learning with Errors).

SNARGs. *Succinct non-interactive arguments* (SNARGs) allow to non-interactively verify **NP** statements with much lower complexity than required for classical **NP** verification; in fact, with complexity that is *independent* of the **NP** language at hand. Being subject to black-box lower bounds, previously constructed SNARGs were either proven secure in the *random oracle model*, known as Micali’s CS proofs, or based on quite elaborate assumptions tailored to specific schemes. In addition, they do not achieve *public-verifiability* and suffer from significant prover overhead.

Here we present two main contributions. First, we present a construction of SNARGs of knowledge (SNARKs), based on a new general notion of *extractable collision-resistant hash function* (ECRH). We further show that ECRHs are essentially necessary for SNARKs, and also give a candidate construction under the knowledge of exponent assumption.

Second, we show *any* SNARK can be “bootstrapped” to obtain a *fully-succinct* SNARK, without expensive preprocessing. Furthermore, in the resulting SNARK that prover time and space complexity is essentially the same as that required for classical **NP** verification. By applying our transformation to known publicly-verifiable SNARKs with preprocessing, we obtain the first publicly-verifiable SNARK in the plain model.

Contents

Preface: What is Knowledge?	xiv
1 Introduction	1
1.1 Results	4
1.2 Road-Map to this Thesis	7
2 Preliminaries	9
2.1 Notation and Standard Computational Concepts	9
2.2 Basic Cryptographic Primitives	11
2.3 A Residual Probability Bound	12
3 Unobfuscatable Functions and Resettable Cryptography	14
3.1 Overview	14
3.1.1 Basic Concepts and Motivation	14
3.1.2 Results	16
3.1.3 Obfuscation and Non-Black-Box Simulation	17
3.1.4 Robust Unobfuscatable Functions and Approximate Obfuscation	19
3.1.5 Resettable Protocols from Robust Unobfuscatable Functions . .	20
3.1.6 How to Construct Robust Unobfuscatable Functions	23
3.1.7 More Related Work	27
3.2 Robust Unobfuscatable Functions - Definitions	28
3.2.1 Robustness from Weaker Robustness	31
3.2.2 Robust Unobfuscatable Functions with a Hardcore Secret	35
3.3 A Construction of Robust Unobfuscatable Functions	38
3.3.1 Required PRFs and Encryption	38
3.3.2 The Construction	40

3.3.3	Black-Box Unlearnability	41
3.3.4	Non-Black-Box Learnability	44
3.3.5	More Efficient Extraction from Fully Homomorphic Encryption	54
3.4	Publicly-Verifiable Robust Unobfuscatable Functions	57
3.4.1	Constructing Verifiable Robust Families	59
3.5	From Verifiable Robust Unobfuscatable Functions to Resettable Protocols	64
3.5.1	Definitions	64
3.5.2	A Basic Residual Probability Bound	68
3.5.3	The Base Protocol	69
3.5.4	Security Analysis	69
3.5.5	Resettable Protocols from Minimal Assumptions	81
3.5.6	A 3-Message Simultaneously-Resettable WI Argument of Knowledge	83
3.6	From Resettable-Sound ZK back to Unobfuscatable Functions	86
4	Extractable One-Way Functions	89
4.1	Overview	89
4.1.1	Basic Concepts and Motivation	89
4.1.2	Results	91
4.1.3	Impossibility with respect to Unbounded Auxiliary-Input	92
4.1.4	Constructions with respect to Bounded Auxiliary-Input	95
4.1.5	Zero Knowledge against Verifiers with Bounded Auxiliary-Input	98
4.1.6	More Related Work	101
4.2	Extractable One-Way Functions	102
4.2.1	Generalized Extractable One-Way Functions	105
4.3	From IO to Impossibility of Unbounded-Auxiliary-Input EOWFs	107
4.3.1	Indistinguishability Obfuscation	108
4.3.2	Puncturable PRFs	108
4.3.3	The Impossibility Result	109
4.4	Bounded-Auxiliary-Input Extractable One-Way Functions	114
4.4.1	Non-Interactive Universal Arguments for Deterministic Computations & Delegation	114
4.4.2	Constructions	116

4.5	2-Message and 3-Message Zero Knowledge against Bounded-Auxiliary-Input Verifiers	123
4.5.1	Definition	123
4.5.2	WI Proof of Knowledge with an Instance-Independent First Message	124
4.5.3	1-Hop Homomorphic Encryption	130
4.5.4	Constructions	131
4.6	Black-Box Lower Bounds	138
5	SNARKs: When Knowledge Meets Succinctness	141
5.1	Overview	141
5.1.1	Basic Concepts and Motivation	141
5.1.2	Results	145
5.1.3	Designated-Verifier SNARKs from ECRHs	147
5.1.4	Bootstrapping SNARKs	151
5.1.5	More Related Work	155
5.2	Basic Definitions	156
5.2.1	SNARKs	156
5.3	Extractable Collision-Resistant Hash Functions and Designated-Verifier SNARKs	159
5.3.1	ECRHs	159
5.3.2	Preliminaries for SNARK Construction	165
5.3.3	Construction of SNARKs from ECRHs	167
5.4	Bootstrapping SNARKs	178
5.4.1	Bootstrapping Publicly-Verifiable Preprocessing SNARKs	179
5.4.2	Bootstrapping Designated-Verifier Preprocessing SNARKs	192
5.4.3	Complexity Preservation	198
6	Conclusion and Future Research	202
	Bibliography	203

List of Tables

- 3.1 The sequence of hybrids; the bit b corresponds to the bit commitment C ; the gray cells indicate the difference from the previous hybrid. . . . 85
- 4.1 The sequence of hybrids; the bit b corresponds to the bit commitment $\vec{C}$; the gray cells indicate the difference from the previous hybrid. . . . 128

List of Figures

3.1	A resettably-sound (concurrent) ZK protocol	69
3.2	A resettably-sound (concurrent) ZK protocol from any one-way function	82
3.3	An rWI three-message Argument of Knowledge (implying srWIAOK)	84
4.1	The circuit C_k	111
4.2	The circuit $C_{k_{e^*}, v^*}$	111
4.3	The auxiliary input distribution Z_n	111
4.4	The function f_n	117
4.5	The relation $\mathcal{R}_n^{\mathcal{F}}(f_n(u), u')$	118
4.6	The function f_n	121
4.7	A 3-message WIPOK with instance-independent first message	127
4.8	A 3-message ZK argument of knowledge against verifiers with b -bounded auxiliary-input.	132
4.9	A 2-message ZK argument against verifiers with b -bounded auxiliary input.	136
5.1	A SNARK for NP	170
5.2	The recursive-verification machine $\mathbb{M}^{y, \tau}$	182
5.3	A bounded-space SNARK for read once computations in NP	184
5.4	A fully-succinct SNARK for NP	191
5.5	The recursive-verification machine $\mathbb{M}^{y, c^\tau}$	195
5.6	A bounded-space (designated-verifier) SNARK for read-once computa- tions in NP	201

Preface: What is Knowledge?

Inspired by the Story of Galileo and Kepler

It was in the months following the invention of the first telescopes (early 17th century). Galileo Galilei, who had apparently constructed the strongest ones, was making striking discoveries way faster than he could publish. On July 25th he discovered that Saturn was situated between two smaller companions that always moved together. Wanting to establish his priority of discovery, but lacking the time to write a proper paper, he sent to Johannes Kepler and other fellow physicists a *commitment* of his discovery. Given that modern bit commitment schemes will be invented only centuries later, Galileo decided to use a naive encryption scheme for this purpose. He sampled a random secret permutation σ , and sent

$$\text{com}_G = \sigma(\text{altissimum planetam tergeminum observavi}) = \text{¹ \\ sma isrmil mepoe taleumibunenugttauiras .}$$

Shortly after, Kepler had replied to Galileo and the others that he also had just made a new discovery and sent a commitment of its own

$$\text{com}_K = \text{stiamil oealm seuepmibrunesnugtmata rmuia .}$$

A couple of weeks later, Galileo was ready to decommit and revealed the discovery ‘‘altissimum planetam tergeminum observavi’’ along with the secret permutation σ . Kepler had then also decided to decommit, and to everyone’s astonishment, revealed the *exact same discovery*, together with a proper permutation of his own τ !

¹Translating from Latin: “I have observed the highest of the planets (Saturn) three-formed”.

Galileo, unwilling to believe that this is a coincidence, suggested that Kepler is a cheat. He proposed the following explanation: when Kepler got his commitment com_G , he simply chose a random permutation π of its own, used it to permute com_G , and sent the result $\text{com}_K = \pi(\text{com}_G)$. Then, when Galileo revealed σ , Kepler simply revealed $\tau = \pi \cdot \sigma$. According to Galileo, Kepler had no *knowledge* of what he was actually committing to. Kepler of course denied. Being unable to resolve this conundrum, they ended up writing a joint paper.

The story, which is only true up to the point of Galileo’s commitment,² exemplifies that demonstrating knowledge, while at the same time trying to protect its contents, could pose a serious challenge. On one hand, it was unfair to require of Kepler to reveal the contents of his commitment ahead of time; on the other, it seems that Kepler’s commitment really does not guarantee that he had actually known the discovery in the time of the commitment.

Cryptographers have systematically studied the problem for the past three decades, starting with the introduction of *zero-knowledge proofs of knowledge*, and evolving to many other forms of *knowledge extraction* concepts and techniques. Still, even today, several challenging questions concerning the problem remain open. These questions are at the core of this thesis.

²In the true story [1, 131], Kepler did no such thing, and did not claim a discovery of his own. He simply tried to decipher Galileo’s commitment, and interestingly found a permutation π that led to a different discovery that Kepler himself has made: he had found two of Mars’ moons.

Chapter 1

Introduction

The *knowledge* of a given adversarial program could be a rather elusive concept, especially in cryptography where it often collides with the requirement for privacy. This is well-exemplified by *zero-knowledge proofs* [106], allowing to prove any **NP** statement “ $x \in \mathcal{L}$ ”, so that the verifier does not learn anything from the proof, but the mere correctness of the statement. In such proofs, it is not clear whether, to provide a convincing proof for a true statement $x \in \mathcal{L}$, the prover must actually “know” a corresponding witness (evidently, for the proof to be zero-knowledge, its transcript does not include a witness in the clear). In fact, it is not clear how to capture what *knowing a witness* means.

In the context of zero knowledge, the study of this question resulted in the concept of *zero knowledge proofs of knowledge* [20, 77]. In such proofs, we require that the proof is not only sound, but also that given a convincing prover for the fact that $x \in \mathcal{L}$, it is possible to *efficiently extract* from its algorithm a corresponding witness. This approach of demonstrating adversarial knowledge by efficient extraction has become a common paradigm, and an inseparable part of cryptographic security analysis. Indeed, quite often *security reductions* or *simulators* rely on an essential step of knowledge extraction from the adversary. For example, in multi-party computation [99], such knowledge extraction is invoked to exhibit both correct behavior of dishonest parties, and privacy of inputs for honest parties.

Black-box vs. non-black-box extraction. Traditionally, the basic technique for extracting knowledge from an adversary is to run it on multiple related inputs to deduce what it “knows” from the resulting outputs. The power of this technique (often called

rewinding) is in that it treats the adversary as a black-box without knowing anything regarding its “internals”. (In some cases, it even allows to capture the knowledge of unbounded algorithms, letting the efficient extraction procedure treat them as oracles.)

However, starting from the work of Goldreich and Krawczyk [98], a number of impossibility results for black-box reductions and simulation show that extraction by rewinding is also limited, and is insufficient for many applications. For a long time, surpassing black-box lower-bounds was considered to be an inconceivable task, and these were basically treated as absolute lower bounds. This was soon shown to be a misconception. In his breakthrough work [11], Barak developed techniques that use the actual adversarial program in an essential way, rather than only the adversary’s input-output functionality. Relying on these techniques, Barak and subsequent works show how to circumvent some of the known black-box lower bounds for zero-knowledge protocols, such as public-coin constant-round protocols, and various kinds of resettably-secure protocols (further discussed below).

A main limitation shared by both classical rewinding techniques and Barak’s non-black-box technique is that they require substantial interaction with the adversary. In particular, there exist rather desirable applications where, not only do there exist black-box lower-bounds [91, 98], but that also fall out of the reach of Barak’s non-black-box techniques. Indeed, these are applications where minimal interaction is an essential requirement, such as 3-message zero-knowledge, where the verifier sends only a single message in the protocol, and succinct non-interactive arguments.

Knowledge assumptions and non-explicit extraction. Damgård [67] proposed an alternative approach to knowledge extraction in the form of the *knowledge of exponent assumption (KEA)*. The assumption essentially states that it is possible to extract the secret value x from any program that, given two random generators (g, h) of an appropriate group G , outputs a pair of group elements of the form g^x, h^x . This approach was then abstracted by Canetti and Dakdouk [51, 52] who formulated a notion of *extractable functions*. These are function families $\{f_e\}$ where, in addition to standard hardness properties, such as one-wayness or collision-resistance, any (possibly adversarial) program $\mathcal{A}$ that given e outputs v in the image of f_e has an “extractor” $\mathcal{E}$ that given e and the code of $\mathcal{A}$, outputs a preimage of v .

Extractable functions provide an alternative (albeit non-explicit) “extraction

method” that does not rely on interaction with the adversary. As an expression of the method’s power, KEA [22, 113], or even general extractable one-way functions [28, 52], are known to suffice for constructing 3-message zero-knowledge protocols. KEA had also led to relatively efficient CCA constructions [22, 67].

The black-box impossibility of some of the above applications imply that it is impossible to obtain extractable functions where the extractor uses the adversary’s program $\mathcal{A}$ only as a black box. Coming up with the suitable non-black-box techniques has been the main obstacle in constructing extractable functions, and previously, no construction with an explicit extraction procedure was known. Instead, for all the existing candidate constructions of extractable functions (e.g., [52, 67]), the existence of such an extractor is merely *assumed*. In particular, any security reduction (or simulator) based on such a knowledge assumption essentially has a “hole” in its code, a sub-routine which we do not know how to explicitly implement, but only believe it exists.

While, this may be better than simply *assuming that a given scheme is secure*, such assumptions are arguably not satisfying. For one, they do not qualify as “efficiently falsifiable” [134]; that is, unlike standard assumptions, here it may not be possible to algorithmically test whether a given adversary breaks the assumption. In addition, the impossibility of extractable functions with black-box extraction only further decreases our confidence in such assumptions, as our current understanding of non-black-box techniques and their limitations is at its infancy.

The tension between non-black-box extraction and obfuscation. Code obfuscation is aimed at making code unintelligible while preserving its functionality, and has been long considered to be a holy grail of cryptography, with diverse and far reaching applications. Barak et al. [16] initiated the rigorous treatment of obfuscation, formulating a number of definitions of security for the task. Until recently, the study of obfuscation produced mainly lower-bounds, and few constructions for very simple classes of programs. This state of affairs changed with the work of Garg et al. [83] who proposed a candidate obfuscator for *general* programs based on multilinear maps [66, 81]. They conjectured that their construction satisfies a rather relaxed notion of *indistinguishability obfuscation* (IO) [16], for which no lower bounds are known. As evidence for the security of their construction, it (and variants thereof) were shown secure in an idealized algebraic model [13, 47].

At an intuitive level, the concept of obfuscation seems to stand in clear contrast with the concept of non-black-box extraction. Indeed, code obfuscation is exactly meant to prevent extraction of any secrets from the obfuscated code, except for those that can be extracted in a black-box manner. In fact, according to the *virtual black-box* paradigm [16], access to an “ideally obfuscated” adversarial code would amount black-box access to the adversary, where black-box lower-bounds apply. Fortunately (at least from the perspective of knowledge extraction), Barak et al. [16] show that virtual black-box obfuscation is impossible in general. The exact tradeoff between what is possible in obfuscation and non-black-box extraction seems far from being well understood.

1.1 Results

In this thesis, we develop new techniques for non-black-box knowledge extraction from adversarial code, both in interactive and non-interactive settings. We demonstrate the power of these techniques in several new cryptographic schemes and protocols achieving features that were previously out of reach. In the process, we further the understanding of the tradeoff between code obfuscation and non-black-box extraction: we prove new lower bounds on obfuscation, and translate them to positive results on extraction, and translate recent positive results for obfuscation to lower-bounds on extraction.

We now describe at high-level each of these results. A more detailed overview of the relevant concepts and techniques is provided in the beginning of each chapter.

Unobfuscatable functions and applications to resettable cryptography.

Barak et al’s impossibility result for virtual black-box obfuscation demonstrates the existence of *unobfuscatable function families*; these are essentially functions where *any* code representation leaks certain information that is not leaked given only black-box access to the function. Relying on unobfuscatable functions, we develop a new non-black-box extraction technique leading to new zero-knowledge protocols with resilience to various forms of resetting attacks [15, 53]. Such protocols can withstand adversaries that may physically reset honest parties to their initial state, forcing them to repeat the protocol’s execution from the same initial state and random tape, whereas the adversary may use different inputs and messages. This setting is motivated by scenarios where cryptographic protocols are run by parties that cannot regenerate fresh randomness or keep a state between different executions, occurring after different reset

attempts. Examples include: parties implemented on stateless hardware, inside virtual machines, or parties that are required to engage in multiple consistent executions in a distributed environment.

Resetable protocols are subject to known lower-bounds for black-box simulation [15, 98]. The simulator in our protocols relies on our non-black-box extraction technique, and thus circumvents these lower-bounds.

We also prove a new stronger impossibility result for a relaxed notion of virtual black-box obfuscation *with approximate functionality*, demonstrating the existence of certain *robust* unobfuscatable functions. Relying on robust unobfuscatable functions results in simplified resettable protocols with improved features in terms of round-complexity, and computational assumptions (comparing to the resettable protocols relying on the Barak et al. (non-robust) unobfuscatable functions, and other existing protocols) . This includes:

- Four-message resettably-sound zero-knowledge protocols, assuming fully-homomorphic encryption. Such protocols guarantee soundness even against a malicious prover that can reset the honest verifier.
- Six-message resettably-sound zero-knowledge protocols, assuming one-way functions.
- Simultaneously-resettable zero-knowledge protocols (with polynomially many rounds), assuming one-way functions. Such protocols can withstand not only a resetting prover, but also malicious verifiers that can reset the honest prover.
- Three-message simultaneously-resettable witness-indistinguishable arguments of knowledge.

The non-black-box simulation based on our extraction technique is substantially different from that of Barak [11], and in particular avoids any reliance on heavy tools such as the PCP theorem [7–10, 75]. One prominent feature of the technique is its compatibility with rewinding techniques from classic black-box zero-knowledge protocols. Indeed, the combination of rewinding in non-black-box simulation has proven instrumental in coping with challenging goals, such as zero-knowledge protocols that are resilient to resetting attacks from both the side of the prover or the verifier. While previous works relied on tailored modifications to Barak’s technique, we give a gen-

eral recipe that allows to combine our non-black-box technique with existing rewinding techniques in a rather simple manner.

On the existence of extractable functions. Like Barak’s non-black-box technique, the extraction technique based on unobfuscatable functions is still only useful in settings with enough interaction, and does not seem to reach the same applications as extractable functions do. The problem of extractable functions is tackled directly in the second part of this thesis. Here we make two headways:

On the negative side, we show that, assuming indistinguishability obfuscation for a certain class of circuits, there do not exist *extractable one-way functions* with a *universal* extractor that works for any adversarial code with any auxiliary-input of unbounded polynomial length.

On the positive side, for adversarial programs with bounded auxiliary-input (and unbounded polynomial running time), we give the first construction of extractable one-way functions *with an explicit extraction procedure*, based on relatively standard assumptions (e.g., sub-exponential hardness of Learning with Errors). We then use these functions to construct the first 2-message zero-knowledge arguments and 3-message zero-knowledge arguments of knowledge, against adversarial verifiers with bounded auxiliary-input, from essentially the same assumptions.

Succinct non-interactive arguments. The third part of this thesis is devoted to a powerful application of non-interactive knowledge extraction - constructing *succinct non-interactive arguments* (SNARGs). SNARGs allow to non-interactively verify **NP** statements with much lower complexity than required for classical **NP** verification; in fact, with complexity that is *independent* of the **NP** language at hand.

As a natural complexity-theoretic problem, SNARGs have been pursued for over two decades. Traditionally, the main focus has been on communication complexity (namely, how short can proofs for **NP** be). In the past few years, however, due to the growing interest in the problem of verifiably delegating computation, the study has extended to other efficiency measures: most importantly, the verifier’s running time, but also prover running time and space complexity (which typically would affect the cost of delegating a given computation). Being subject to black-box lower bounds [91], previously constructed SNARGs were either proven secure in the *random oracle model*, known as Micali’s CS proofs [128], or based on quite elaborate assumptions tailored to

specific schemes [71].

Our first contribution, in this context, is a construction of SNARGs, based a new general notion of *extractable collision-resistant hash function* (ECRH). The construction, in fact, yields SNARGs of knowledge (SNARKs), where it is possible to extract a witness for the proven statement from any convincing prover. Such witness extraction is often required when composing SNARGs with other cryptographic primitives, or with themselves (as we shall see later on). We further show that ECRHs are essentially necessary for SNARKs, and also give a candidate construction under the knowledge of exponent assumption.

The latter SNARKs construction, as well as previous SNARG constructions in the plain model, fall short of attaining certain desirable features. First, these constructions all require that the verifier keeps a private verification state. *Publicly-verifiable SNARKs* are only known either in the random oracle model, or in a model that allows expensive offline preprocessing. Second, known SNARGs require from the prover significantly more time or space than required for classical NP verification.

We show that, assuming collision-resistant hashing, *any* SNARK can be “bootstrapped” to obtain a *fully-succinct* SNARK, without expensive preprocessing. Furthermore, the resulting SNARK is *complexity-preserving* in the sense that prover time and space complexity is essentially the same as that required for classical **NP** verification. By applying our transformation to known publicly-verifiable SNARKs with preprocessing, we obtain the first publicly-verifiable SNARK in the plain model. At the heart of our transformation is a new technique for *recursive composition* of SNARKs.

1.2 Road-Map to this Thesis

In Chapter 3, we present the results on unobfuscatable functions and resettable protocols. In Chapter 4, we present the result on the existence of extractable functions. In Chapter 5, we present our result on succinct non-interactive arguments.

Each chapter starts with an overview of the relevant concepts, the results, and techniques. The chapters are mostly self-contained, with occasional references to each other, and can in principle be read in any order.

The chapters are based on the following publications:

- Chapter 3 is based on:

- *From the impossibility of obfuscation to a new non-black-box simulation technique.* Bitansky and Paneth. FOCS 2012 [35].
 - *On the impossibility of approximate obfuscation and applications to resettable cryptography.* Bitansky and Paneth. STOC 2013 [36].
 - Unified journal version to appear in SIAM journal on computing.
- Chapter 4 is based on:
 - *On the existence of extractable one-way functions.* Bitansky, Canetti, Paneth, and Rosen. STOC 2014 [32]. Journal version in submission.
 - Chapter 5 is based on:
 - *From extractable collision-resistance to succinct non-interactive arguments of knowledge, and back again,* Bitansky, Canetti, Chiesa, and Tromer, ITCS 2012 [29]. Journal version joint with Goldwasser, Lin, and Rubinfeld [28] in submission.
 - *Recursive composition and bootstrapping for SNARKs and proof-carrying data.* Bitansky, Canetti, Chiesa, and Tromer. STOC 2013 [30].

Chapter 2

Preliminaries

In this chapter, we review basic notation and definitions that reoccur throughout the thesis. Further background on these basic concepts can be found in [93, 94].

2.1 Notation and Standard Computational Concepts

Functions and asymptotics. We use standard notation for functions. We denote by $f : X \rightarrow Y$ a function that maps elements in a domain set X to a range set Y . We denote by $\text{Image}(f) := f(X) \subseteq Y$ the image set $\{y \in Y : \exists x \in X, f(x) = y\}$. For $y \in Y$, we denote by $f^{-1}(y)$ the set $\{x \in X : f(x) = y\}$ of preimages of y . We use standard O -notations O, o, Ω, ω to denote order of growth of functions. A function $\nu : \mathbb{N} \rightarrow [0, 1]$ is said to be negligible if $\nu(n) = n^{-\omega(1)}$, i.e. ν decays faster than any polynomial. We say that an event occurs with overwhelming probability if it occurs with probability $1 - \nu(n)$ for some negligible function ν . We often denote by negl an unspecified negligible function (e.g., when we say that for all $n \in \mathbb{N}$ and event A_n occurs with probability at most $\text{negl}(n)$, we mean that this is the case for some negligible function). We say that a function $\varepsilon : \mathbb{N} \rightarrow [0, 1]$ is noticeable if $\varepsilon(n) = n^{-O(1)}$. We denote by $\text{poly}(n)$ an unspecified polynomial.

Distributions. For a distribution D , we denote by $\text{supp}(D)$ the support of D ; namely, the set of elements to which D assigns a non-zero probability. We denote by $x \leftarrow D$ the process of sampling an element x from $\text{supp}(D)$ according to the distribution D . Abusing notation, for a finite set S , we denote by $x \leftarrow S$ the process of sampling x from S uniformly at random. We denote by U_n the uniform distribution over bit strings in

$\{0, 1\}^n$.

Models of computation. We rely on the standard notions of Turing machines and Boolean circuits. We say that a (uniform) Turing machine is PPT if it is probabilistic and runs in polynomial time. A polynomial-size (or just polysize) circuit family $\mathcal{C}$ is a sequence of circuit $\mathcal{C} = \{C_n\}_{n \in \mathbb{N}}$, such that each circuit C_n is of polynomial size $n^{O(1)}$ and has $n^{O(1)}$ inputs and outputs bits. We shall often refer to Turing machines $\mathcal{M}^f$ or circuits C^f that make black-box use of some oracle function f .

Adversaries. We follow the standard habit of modeling any efficient adversary strategy as a family of polynomial-sized circuits. For an adversary $\mathcal{A}$ corresponding to a family of polysize circuits $\{\mathcal{A}_n\}_{n \in \mathbb{N}}$, we often omit the subscript n , when it is clear from the context. In Chapter 4, we will also consider uniform adversaries, i.e. PPT adversaries, or more generally PPT adversaries with bounded non-uniform advice. Such adversaries are given by a PPT Turing machine that for input size n also get advice of size $b(n)$ for some fixed bound b . The running time of such adversaries may still be any polynomial, in particular, it may be larger than $b(n)$.

Computational indistinguishability. A distribution ensemble $\mathcal{D}$ is an infinite sequence of distributions $\{\mathcal{D}_i\}_{i \in I}$, where each distribution is indexed by some i taken from some infinite set of string $I \subseteq \{0, 1\}^*$, and is defined over $\{0, 1\}^{\text{poly}(|i|)}$ for some fixed polynomial $\text{poly}(\cdot)$. We say that two ensembles $\mathcal{D}, \mathcal{D}'$ over the same index set I are computationally indistinguishable if for any polysize distinguisher $\mathcal{A} = \{\mathcal{A}_i\}_{i \in I}$, and every $i \in I$,

$$\left| \Pr_{d \leftarrow \mathcal{D}_i} [\mathcal{A}(d) = 1] - \Pr_{d' \leftarrow \mathcal{D}'_i} [\mathcal{A}(d') = 1] \right| \leq \text{negl}(|i|) .$$

We denote this by $\mathcal{D} \approx_c \mathcal{D}'$.

Languages and relations. For a binary relation $\mathcal{R} \in \{0, 1\}^* \times \{0, 1\}^*$, we denote by $\mathcal{L}_{\mathcal{R}} = \{x : \exists w \text{ such that } (x, w) \in \mathcal{R}\}$. For $(x, w) \in \mathcal{R}$, we say that w is a witness to the fact that $x \in \mathcal{L}_{\mathcal{R}}$. We denote by $\mathcal{R}(x)$ the set $\{w : (x, w) \in \mathcal{R}\}$ of witnesses for $x \in \mathcal{L}_{\mathcal{R}}$. $\mathcal{R}$ is said to be an **NP** relation if there exists a (uniform) Turing machine $\mathcal{M}_{\mathcal{R}}$ such that $\mathcal{M}(x, w)$ runs in time $\text{poly}(|x|)$ for some fixed polynomial $\text{poly}(\cdot)$ and $\mathcal{M}(x, w) = 1$ iff $(x, w) \in \mathcal{R}$. $\mathcal{L}$ is said to be an **NP** language if $\mathcal{L} = \mathcal{L}_{\mathcal{R}}$ for some **NP** relation $\mathcal{R}$.

We say that a language $\mathcal{L} \subseteq \{0, 1\}^*$ is in **P** if there exists a polytime (uniform) Turing machine $\mathcal{M}$ that decides $\mathcal{L}$, i.e., $\mathcal{L} = \{x : \mathcal{M}(x) = 1\}$. For a function $T : \mathbb{N} \rightarrow \mathbb{N}$, and a language $\mathcal{L} \subseteq \{0, 1\}^*$ we write that $\mathcal{L} \in \mathbf{Dtime}(T)$ if there is a (uniform) Turing machine that runs in time at most $T(|x|)$ on any input x and decides $\mathcal{L}$.

2.2 Basic Cryptographic Primitives

One-way functions. One-way functions are functions that are easy to compute, but hard to invert.

Definition 1 (OWF). *$f : \{0, 1\}^* \rightarrow \{0, 1\}^*$ is a one-way function if it is computable in polynomial time, and for every polysize adversary $\mathcal{A}$, and every $n \in \mathbb{N}$,*

$$\Pr_{x \leftarrow \{0, 1\}^n} [\mathcal{A}(f(x)) \in f^{-1}(f(x))] \leq \text{negl}(n) .$$

Pseudorandom generators. Pseudorandom generators take as input a short seed and expand it into a longer string that is computationally indistinguishable from a truly uniform string.

Definition 2 (PRG). *Let $e(n)$ be a polytime computable function. A polytime computable function $G : \{0, 1\}^* \rightarrow \{0, 1\}^*$ is a pseudo random generator with expansion e , if for any $s \in \{0, 1\}^n$, $|G(s)| = e(n) > n$, and*

$$\{G(U_n)\}_{n \in \mathbb{N}} \approx_c \{U_{e(n)}\}_{n \in \mathbb{N}} .$$

Pseudorandom functions. An efficiently computable family of functions is pseudorandom if oracle access to a random function in the family is computationally indistinguishable from oracle access to a truly random function.

Definition 3 (PRF). *Let $\mathcal{PRF} = \{\text{PRF}_k\}_{k \in \{0, 1\}^n, n \in \mathbb{N}}$ be a family of functions where for each $s \in \{0, 1\}^n$, $\text{PRF}_k : \{0, 1\}^{\ell(n)} \rightarrow \{0, 1\}^{\ell'(n)}$, for some polynomially bounded efficiently computable length functions ℓ, ℓ' , and there is an efficient algorithm that given $k \in \{0, 1\}^n, x \in \{0, 1\}^{\ell}$ computes $\text{PRF}_k(x)$. Let $\mathcal{R}_n$ denote the set of all functions $\{0, 1\}^{\ell(n)} \rightarrow \{0, 1\}^{\ell'(n)}$. We say that $\mathcal{PRF}$ is a pseudo random if for polysize*

distinguisher $\mathcal{D}$, it holds that

$$\left| \Pr_{k \leftarrow \{0,1\}^n} [\mathcal{D}^{\text{PRF}_k}(1^n)] - \Pr_{R \leftarrow \mathcal{R}_n} [\mathcal{D}^R(1^n)] \right| \leq \text{negl}(n) .$$

Theorem 4 ([95, 116]). *If there exist one-way functions, then there exists PRGs and PRFs.*

The Goldreich-Levin theorem. We shall rely on the following classic theorem by Goldreich and Levin.

Theorem 5 ([92]). *There exists a PPT oracle aided (decoding) algorithm $\mathcal{D}$ satisfying the following. Let $x \in \{0,1\}^n$ and let $\mathcal{O}_x(\cdot)$ be an oracle with $1/2 + \varepsilon$ agreement with the inner product function $\langle x, r \rangle \bmod 2$. Then $\Pr [\mathcal{D}^{\mathcal{O}_x(\cdot)}(1^n, 1^{\varepsilon^{-1}}) = x] \geq \text{poly}(\varepsilon, n^{-1})$.*

We deduce the following direct corollary which we shall rely on.

Corollary 6 (special case). *There exists a PPT inverter $\mathcal{I}$, such that for any function f , string $k \in \{0,1\}^n$, and oracle-aided distinguisher $\mathcal{A}$, if $\mathcal{A}^f$ distinguishes $(r, \langle k, r \rangle)$ from (r, b) , for a random $r \leftarrow \{0,1\}^n$ and $b \leftarrow \{0,1\}$ with probability ε , then $\mathcal{I}^f(1^n)$ obtains k with probability $\text{poly}(\varepsilon, n^{-1})$.*

2.3 A Residual Probability Bound

In this section, we prove a basic probability bound that will be used a couple of times in Chapter 3.

Let $S_1, \dots, S_m$ be random variables (intuitively, describing an m -step process), and let $G_1, \dots, G_m$ be events where G_i is determined by S_i alone. Denote by H_i the event that the first $i - 1$ stages $S_1, \dots, S_{i-1}$ are such that the event G_i occurs with bounded probability δ (when S_i is sampled conditioned on $S_1, \dots, S_{i-1}$); namely, H_i is the event that $S_1, \dots, S_{i-1}$ satisfy:

$$\Pr_{S_i | S_1, \dots, S_{i-1}} [G_i | S_1, \dots, S_{i-1}] \leq \delta .^1$$

The following intuitive claim shows an exponential decay in the probability that k events G_i occur when the residual probability of their occurrence is bounded through

¹More explicitly, we mean that H_i is that even that $(S_1, \dots, S_{i-1}) = (s_1, \dots, s_{i-1})$, such that $(s_1, \dots, s_{i-1})$ satisfy $\Pr_{S_i | (S_1, \dots, S_{i-1}) = (s_1, \dots, s_{i-1})} [G_i | (S_1, \dots, S_{i-1}) = (s_1, \dots, s_{i-1})] \leq \delta$.

every stage of the process.

Claim 2.3.1. *For every $\{i_1, i_2, \dots, i_k\} \subseteq [m]$:*

$$\Pr_{S_1, \dots, S_m} \left[\left(\bigwedge_{j=1}^k G_{i_j} \right) \wedge \left(\bigwedge_{j=1}^m H_j \right) \right] \leq \delta^k .$$

Proof. First we show by induction that for any $0 \leq \ell \leq k$:

$$\Pr \left[\left(\bigwedge_{j=1+k-\ell}^k G_{i_j} \right) \wedge \left(\bigwedge_{j=1+k-\ell}^k H_{i_j} \right) \mid \left(\bigwedge_{j=1}^{k-\ell} G_{i_j} \right) \wedge \left(\bigwedge_{j=1}^{k-\ell} H_{i_j} \right) \right] \leq \delta^\ell$$

For $\ell = 0$ the claim clearly holds. Assuming the claim holds for $\ell - 1$ we have:

$$\begin{aligned} & \Pr \left[\left(\bigwedge_{j=1+k-\ell}^k G_{i_j} \right) \wedge \left(\bigwedge_{j=1+k-\ell}^k H_{i_j} \right) \mid \left(\bigwedge_{j=1}^{k-\ell} G_{i_j} \right) \wedge \left(\bigwedge_{j=1}^{k-\ell} H_{i_j} \right) \right] \\ & \leq \Pr \left[\left(\bigwedge_{j=1+k-\ell}^k G_{i_j} \right) \wedge \left(\bigwedge_{j=2+k-\ell}^k H_{i_j} \right) \mid \left(\bigwedge_{j=1}^{k-\ell} G_{i_j} \right) \wedge \left(\bigwedge_{j=1}^{1+k-\ell} H_{i_j} \right) \right] \\ & \leq \Pr \left[G_{i_{1+k-\ell}} \mid \left(\bigwedge_{j=1}^{k-\ell} G_{i_j} \right) \wedge \left(\bigwedge_{j=1}^{1+k-\ell} H_{i_j} \right) \right] \cdot \\ & \Pr \left[\left(\bigwedge_{j=2+k-\ell}^k G_{i_j} \right) \wedge \left(\bigwedge_{j=2+k-\ell}^k H_{i_j} \right) \mid \left(\bigwedge_{j=1}^{1+k-\ell} G_{i_j} \right) \wedge \left(\bigwedge_{j=1}^{1+k-\ell} H_{i_j} \right) \right] \\ & \leq \Pr \left[G_{i_{1+k-\ell}} \mid \left(\bigwedge_{j=1}^{k-\ell} G_{i_j} \right) \wedge \left(\bigwedge_{j=1}^{1+k-\ell} H_{i_j} \right) \right] \cdot \delta^{\ell-1} \\ & \leq \delta^\ell , \end{aligned}$$

where the first to last inequality is by the induction hypothesis and the last inequality is due to the fact that the event $\left(\bigwedge_{j=1}^{k-\ell} G_{i_j} \right) \wedge \left(\bigwedge_{j=1}^{1+k-\ell} H_{i_j} \right)$ is contained in the event $H_{i_{1+k-\ell}}$. By setting $\ell = k$ we get that

$$\Pr \left[\left(\bigwedge_{j=1}^k G_{i_j} \right) \wedge \left(\bigwedge_{j=1}^m H_j \right) \right] \leq \Pr \left[\left(\bigwedge_{j=1}^k G_{i_j} \right) \wedge \left(\bigwedge_{j=1}^k H_{i_j} \right) \right] \leq \delta^k .$$

□

Chapter 3

Unobfuscatable Functions and Resetable Cryptography

In this chapter, we present a new impossibility result for *obfuscation with approximate functionality*. Based on this result, we develop a new non-black-box extraction technique, which in turn leads to a new (non-black-box) zero-knowledge simulation technique. We demonstrate the power of the technique by exhibiting various new resettable-secure protocols with improved round complexity and weaker computational assumptions. The chapter is based on [35, 36].

3.1 Overview

In this section, we give a high-level overview of the results in this chapter, and of the corresponding techniques. Before that, we recall the basic relevant concepts and main motivation, which were briefly described in the introduction.

3.1.1 Basic Concepts and Motivation

Zero-knowledge (ZK) protocols [106] are a cornerstone of modern cryptography; they can express all **NP** computations [100], and are essential to almost any form of secure computation.

The ZK guarantee of a protocol is established by exhibiting an efficient *simulator* that can simulate the view of any malicious verifier from the verifier’s code and the statement alone. Following the common practice of black-box reductions, the first ZK protocols all relied on simulators that only use the verifier as a black-box, without

making any explicit use of its code. However, while sufficient for a variety of powerful applications, ZK protocols with black-box simulation were soon shown to have inherent limitations. A known example is the impossibility of constant-round public-coin ZK [98]. Surpassing such black-box impossibilities was considered to be an inconceivable task.

This barrier was crossed with the ground-breaking result of Barak [11] that introduced a non-black-box simulation technique. Barak’s technique allowed, in particular, to achieve constant-round public-coin ZK. Subsequently, the technique was utilized to achieve various cryptographic goals, most of which were previously limited by black-box impossibilities, e.g., [12, 15, 17, 19, 54, 69, 107–110, 138–141].

Today, an impressive variety of protocols rely on non-black-box simulation; however, the common base of all of these protocols is the very same technique of Barak. A main question addressed in this chapter is

Q : *Is Barak’s technique inherent in non-black-box simulation? can black-box impossibilities may be circumvented using different techniques and tools?*

. Beyond improving our understanding of non-black-box simulation, answering the above question may also lead to protocols with improved features.

Resettable soundness - the bottleneck in black-box impossibilities for ZK.

Aiming to make progress on this question, we focus on a specific setting that seems fundamental in ZK black-box impossibilities - the setting of *resettablely-sound zero-knowledge* [15, 129]. Indeed, many of the known black-box ZK impossibilities can be derived by a reduction to the impossibility of resettablely-sound ZK with a black-box simulator [15, 98, 142].

A ZK argument is resettablely-sound if it remains sound, even if an adversarial prover may reset the honest verifier to its initial state and random tape, and repeat the interaction in any way it chooses (equivalently, the verifier may be rewound to any previous state). As observed by Barak et al. [15], in resettablely-sound protocols, the ZK requirement cannot be fulfilled by a black-box simulator (except for trivial languages). Intuitively, the difficulty is that a cheating prover may execute a resetting attack that emulates the strategy of the black-box simulator, and thus violate soundness. In contrast, a resettablely-sound ZK protocol was constructed by Barak et al. [15] using Barak’s non-black-box simulation technique.

The difficulty in extending Barak’s technique. Attempting to extend [15] to resist more elaborate types of resetting attacks (e.g., *simultaneous resetting*), one encounters significant technical difficulties. To overcome these difficulties, subsequent works have extended Barak’s technique in various ways [54, 58, 63, 69, 107, 109, 110]. A common theme in these works is to extend Barak’s technique so it can be combined with rewinding techniques. While allowing to leverage powerful black-box simulation techniques (e.g., [146]), they require tailored modifications to Barak’s technique, and result in rather complicated protocols.

3.1.2 Results

We present a new non-black-box simulation technique that is fundamentally different from Barak’s technique. Using our technique we give new protocols for resettable-sound zero-knowledge and related tasks.

The simulation in these protocols uses our new technique as well as rewinding techniques. In contrast to the tailored modifications of Barak’s technique in previous work, we give a general recipe for combining our technique with rewinding. This yields simplified resettable protocols for various tasks, as well as improvements in round complexity and required computational assumptions. Concretely, we show:

Theorem 7 (informal).

1. *Assuming fully-homomorphic encryption, there exists a four-message resettable-sound zero-knowledge protocol. (Previous protocols required at least six messages.)*
2. *Assuming one-way functions, there exist*
 - *six-message resettable-sound zero-knowledge protocols. (An eight-message protocol, based also on one-way functions, was demonstrated by Chung, Pass, and Seth [63] shortly before ours; we provide an alternative construction. Previous constructions relied on collision-resistant hashing [15].)*
 - *simultaneously-resettable zero-knowledge protocols (with polynomially many rounds). (Previous protocols relied on collision-resistant hash functions and trapdoor permutations [69].)*
3. *Assuming trapdoor permutations, there exist a three-message simultaneously-resettable witness-indistinguishable argument of knowledge. (Previous protocols*

relied also on collision-resistant hashing and required at least ten messages [58].)

In the heart of our technique, is a new method for extracting a short trapdoor from adversarial code. The extraction technique, in fact, follows from a new impossibility result for general program obfuscation that extends the impossibility result of Barak et al. [16] to the case of obfuscation with approximate functionality; thus, settling a question left open by Barak et al..

Theorem 8 (informal). *Assuming trapdoor permutations, there exist functions that cannot be obfuscated, even if the obfuscation is only required to approximate the original functionality.*

Finally, we show that any resettably-sound ZK argument can be transformed into a family of functions that cannot be obfuscated, thereby establishing a two-way connection between the impossibility of obfuscation and non-black-box simulation in resettable protocols.

We next elaborate on each of the results and the techniques used to achieve it.

3.1.3 Obfuscation and Non-Black-Box Simulation

The problem of *program obfuscation* concerns the task of rewriting programs in a way that makes their code “unintelligible”, without destroying its functionality. The rigorous study of the problem was initiated in the work of Barak et al. [16], which formalized secure obfuscation according to the *virtual black-box* notion. At high-level, this notion requires that whatever an efficient learner can deduce, given an obfuscation $\tilde{P}$ of a program P , should also be learnable, given only black-box access to P . The same work shows, however, that for some programs, this notion is not achievable. Concretely, [16] show that there exists an *unobfuscatable* family of functions $\{f_k\}$ for which *any* program $\tilde{P}$ that computes a function f_k leaks the key k ; that is, k can be efficiently *extracted* from $\tilde{P}$ ’s code. However, given only black-box access to f_k for a randomly chosen key k , the key k cannot be learned.

At an intuitive level, unobfuscatable functions suggest a meaningful way to use the code of the adversarial verifier in non-black-box simulation. Following this intuition, we design a protocol where the verifier evaluates an unobfuscatable function f_k . A non-black-box simulator would be able to learn k from the code of any malicious verifier

and use it to simulate. In contrast, a resetting prover, like a black-box simulator, would only get black-box access to the verifier and would not learn k .

A naive protocol and the problem of non-robust extraction. We now describe a naive resettable-sound ZK protocol based on the intuition presented above. Following the commonly used Feige-Lapidot-Shamir paradigm [76], the protocol proceeds in two phases: the first phase defines a trapdoor which is known by verifier but is “hard to compute” for the prover, and in the second phase, the prover gives a witness indistinguishable (WI) proof that either the statement is correct or that it “knows” the trapdoor. In our case, to obtain resettable-soundness we will have to ensure that the trapdoor is hard to compute, even given the ability to reset the verifier. We will also use a resettable-sound WI proof in the second phase of the protocol (such proofs follow directly from classical protocols [78]). For ZK, we will exhibit a simulator that, given the code of the verifier, can extract the trapdoor and successfully complete the simulation.

As suggested above, the trapdoor in our protocol will be a key k for an unobfuscatable function f_k chosen at random by the verifier. In the trapdoor generation phase of the protocol, the prover asks the verifier to evaluate the function f_k on an input x of his choice (for simplicity of exposition, we assume that this interaction indeed determines the key k and the trapdoor statement for the second phase). To show that the protocol is resettable-sound, we note that any prover that can reset the verifier in the first phase can at most evaluate the function f_k on many inputs x of his choice; however, given such “black-box” access to f_k , it is impossible to learn k and cheat in the second phase. To show that the protocol is ZK, we need to show a simulator that can extract the trapdoor k from the code of any cheating verifier. Assuming that the cheating verifier correctly evaluates the function f_k on any input sent by the prover, the simulator would be able to learn k from the code of the verifier’s next message function. However, if on certain inputs x the verifier deviates from the protocol and does not respond with $f_k(x)$, the extraction guarantee no longer holds and simulation cannot be finished.

3.1.4 Robust Unobfuscatable Functions and Approximate Obfuscation

A first solution to the above problem is to evaluate the function f_k in the first phase of the protocol using a *secure function evaluation* protocol that secure against a resetting provers. Intuitively, in this solution, the secure function evaluation protocol guarantees that the verifier must correctly compute the function on all inputs, or always abort the computation. This solution relies on semi-honest *oblivious transfer*. Guaranteeing that the secure function evaluation protocol is also secure against a resetting prover results in a rather complicated protocol with a large (but constant) number of rounds.

Here we focus on a different approach relying on stronger unobfuscatable functions that provide a robust extraction guarantee. Specifically, it should be possible to extract the key k from any program $\tilde{P}$ that *approximately* computes the function f_k , rather than exactly. More concretely, we define (publicly) verifiable *robust* unobfuscatable functions that have, in addition to the secret key k , an associated public verification key vk . The verification key can be used to verify that a pair (q, a) indeed satisfies $a = f_k(q)$. On one hand, we require that a randomly chosen key k cannot be learned given the verification key vk and black-box access to f_k . On the other hand, we require that it is possible to extract k from vk and any program $\tilde{P}$ that computes a function f_k , *even on some small (but noticeable) fraction of its inputs*. We present a construction of verifiable robust unobfuscatable functions based on ZAPs [74] and non-interactive commitments.

Using verifiable robust unobfuscatable functions restores the security of the naive protocol above. As before, the trapdoor will be a key k for a robust unobfuscatable function f_k chosen at random by the verifier. In the first phase of the protocol, the prover will receive the public verification key vk , and ask the verifier to evaluate the associated function on a *uniformly random* input q . The prover will use vk to verify that f_k was evaluated correctly and abort otherwise. We argue that the revised protocol is ZK: given the code of any cheating verifier, the verifier either makes the honest prover abort with overwhelming probability, in which case simulation is trivial, or the verifier evaluates f_k correctly on some noticeable fraction of inputs. In this case, we can extract k from the verifier's next message function and simulate the second phase.

Impossibility of approximate obfuscation. In addition to their application to resettably-sound ZK, verifiable robust unobfuscatable functions also settle an open question raised by Barak et al. [16] regarding the possibility of *approximate obfuscation*. In approximate obfuscation [16], the obfuscated program $\tilde{P}$ is only required to agree with P on some significant fraction of inputs. By constructing verifiable robust unobfuscatable functions, we give a negative answer to this question.

Constructing robust unobfuscatable functions. Our construction consists of two steps: First, we construct a weaker object that we call (plain) *robust unobfuscatable functions*. Such functions have no verification key, however, they still allow extraction from any program $\tilde{P}$ that only compute f_k on a small fraction of inputs, but only assuming that on all other inputs $\tilde{P}$ outputs a special $\perp$ symbol. We construct such robust unobfuscatable functions from any one-way function. In the second step, we “compile” robust unobfuscatable functions into full fledged verifiable robust unobfuscatable functions based on ZAPs [74] and non-interactive commitments.

Simulation from robust unobfuscatable functions vs. Barak’s simulation.

Both our protocol and Barak’s protocol follow the Feige-Lapidot-Shamir paradigm [76], in which the prover ultimately proves a statement that has a trapdoor witness. A key technical difference between our technique and that of Barak is that our simulator extracts from the verifier’s code a “short” trapdoor of some fixed polynomial length, whereas the simulation of Barak can be seen as using the entire verifier’s code as a “long” trapdoor whose length is not a priori bounded by any fixed polynomial. To facilitate the use of such a long trapdoor, Barak’s protocol involves a WI *universal argument* [14] while our protocols use a standard WI argument. The ability to extract a short trapdoor explains the compatibility between our technique and rewinding techniques, and enables our simplified constructions. Another difference between our technique and Barak’s is that in our technique, the verifier makes inherent use of private coins to sample its trapdoor, a key of an unobfuscatable function.

3.1.5 Resetable Protocols from Robust Unobfuscatable Functions

We next discuss the resettably-secure protocols constructed based on our new technique. As we shall see, compared to known constructions, our protocols require less rounds, weaker assumptions, and are arguably simpler.

All of our protocols follow a similar design paradigm of *resettable slots* that abstracts the use of robust unobfuscatable functions. Commonly used in classic ZK protocols (with black-box simulation), a *slot* is a sub-protocol where the verifier proves knowledge of a trapdoor. While a cheating prover cannot learn the trapdoor from the slot’s execution, the simulator extract the trapdoor by rewinding the verifier during the slot’s execution. The difficulty in using slots within a resettable-sound ZK protocol is that a cheating prover can use resetting to extract the trapdoor and break soundness. The concept of a resettable slot, suggested by [69], requires that the trapdoor cannot be learned by rewinding the slot, but can still be extracted by the non-black-box simulator from the code of a verifier that executes the slot.

Concretely, in our resettable slot, the verifier’s trapdoor is a key k for a verifiable robust unobfuscatable function, which is determined prior to the slot’s execution by sending the verification key vk to the prover. The slot consists of the verifier evaluating f_k on a random input selected by the prover.

Following the resettable slot paradigm, we are able to take ZK protocols with black-box simulation and “add resettable-soundness” in a modular way, replacing (standard) slots with resettable slots. This results in protocols that are conceptually and practically simpler than those based on Barak’s technique [54, 58, 63, 69]. We now overview of the different protocols obtained.

Resettable-sound ZK in four messages. Plugging a resettable slot into the the four-message ZK protocol of Feige and Shamir [77] results in a four-message resettable-sound ZK protocol. Previous constructions relying on Barak’s technique [15] required at least six messages.

Simulating such a protocol with a single slot requires verifiable robust unobfuscatable functions that have a fast enough extraction algorithm. We show how to construct such functions based on *rerandomizable fully homomorphic encryption*. Subsequently to our work, Chung et al. [60] construct a four-message resettable-sound ZK protocol based on one-way functions, relying on a different approach.

Resettable-sound ZK from one-way functions. By relying directly on robust unobfuscatable functions (that are not verifiable), we achieve resettable-sound ZK from the minimal assumption of one-way functions. Our protocol mimics the steps of compiling robust to verifiable robust unobfuscatable functions, but instead of using ZAPs,

and non-interactive commitments, we use interactive (resettable) WI arguments and commitments from one-way functions. Our construction is an alternative to the first construction of resettable-sound ZK from one-way functions by Chung, Pass, and Seth [63] that showed how to replace the collision-resistant hash functions in to protocol of [15] with an interactive use of signatures.

Simultaneously-resettable ZK from one-way functions. The notion of resettable ZK [53] requires that even cheating verifiers that may reset the prover cannot gain knowledge from the protocol. Barak et al. [15] contemplated the existence of protocols that are simultaneously resettable ZK and resettable-sound. Such protocols were first constructed in the influential work of Deng, Goyal, and Sahai [69]. The main step in the construction of [69] is to construct a resettable-sound concurrent ZK protocol. To do so, they present a sophisticated extension of Barak’s technique that carefully combines existing rewinding strategies from concurrent ZK.

We get a simpler construction of resettable-sound concurrent ZK by plugging our resettable slot into the concurrent ZK protocol of Richardson and Kilian [146]. Following the transformation of [69], we get a simultaneously-resettable ZK protocol. By directly using robust unobfuscatable functions (that are not verifiable), we get a resettable-sound concurrent ZK protocol from any one-way function where previous constructions also relied on one-to-one one-way function [63, 69]. Using a construction of simultaneously-resettable WI arguments from one way functions [61], within the transformation of [69], we get a construction of simultaneously-resettable ZK from one-way functions.

Our protocol inherits its polynomial number of rounds from the protocol of Richardson and Kilian. An open question is whether our technique can be combined with state of the art concurrent ZK protocols [122, 144] to reduce the number of rounds to poly-logarithmic. This turns out to depend on the relation between the unobfuscatable function’s extraction time, and the depth of recursive rewinding in these protocols. Concretely, combining these protocols with our techniques requires linear time extraction. We do not know if such unobfuscatable functions exist.

Simultaneously-resettable WI argument-of-knowledge in 3 messages. The simultaneously-resettable ZK protocol of Deng, Goyal and Sahai [69] does not satisfy the desirable proof-of-knowledge property. Cho et al. [58] show a simultaneously-resettable

WI argument-of-knowledge with a large constant number of rounds. This is in contrast to standard constructions of WI arguments-of-knowledge in 3 messages [78]. Using a resettable slot, we construct a 3-message simultaneously-resettable WI argument-of-knowledge. In our construction, the extraction property of the resettable slot is used to by the knowledge extractor to obtain a witness from the prover.

3.1.6 How to Construct Robust Unobfuscatable Functions

The main technical challenge in this chapter is constructing (plain) robust unobfuscatable functions from one-way functions. We now give an overview of the main ideas on which our construction is based.

The starting point for our construction is the construction of Barak et al. [16] of (non-robust) unobfuscatable functions, and can be seen as a “random-self-reducible” version of it. We start by describing the construction of Barak et al., and then move on to describe the main modifications that we introduce.

The construction of Barak et al.. Barak et al. construct a family $\mathcal{F} = \{f_k\}$ of unobfuscatable functions as follows. The key k consists of two random strings (a, b) and a key sk for a symmetric encryption scheme. For a simpler exposition, let us assume for now that the encryption scheme is *fully homomorphic*. Later, we explain how to modify the construction to rely on CCA-1 symmetric encryption (that can be based on one-way functions). For simplicity, we describe $f_k = f_{a,b,\text{sk}}$ as a randomized function; eventually, it is made deterministic using a pseudo-random function. The function is defined as follows:

1. On input a , output b .
2. On input “Encrypt”, output an encryption of a .
3. On any input that is an encryption of b , output b .

Given black-box access to a random function from the family, the only way to learn b is to first learn a , which will break the security of the encryption scheme, or to produce an encryption of b , which is information theoretically impossible, without first querying the function on a . On the other hand, given a circuit C that computes $f_{a,b,\text{sk}}$, we can obtain an encryption of a , evaluate the circuit C homomorphically on this encryption, and obtain an encryption of b . Then, we can learn b using another

black-box application of C . Note that in the above construction we only learn b instead of the entire key of the function. Similarly, in the rest of our constructions we will only be able to extract some unlearnable property $\pi(k)$, rather than the entire key k of the unobfuscatable function; this will be sufficient for all of our applications.

Making the construction “random-self-reducible”. The latter construction is not robust; for example, given a circuit C that computes $f_{a,b,\text{sk}}$ but errs on the single input a , we can no longer extract b . At high-level, we overcome this problem by defining a new function family $\mathcal{G}$ such that $\mathcal{F}$ is randomly reducible to $\mathcal{G}$. That is, for any input q , we can compute $f_{a,b,\text{sk}}(q)$ by evaluating a corresponding $g \in \mathcal{G}$ on inputs that are individually random, but are correlated together according to q . It is important that the new function $\mathcal{G}$ remains black-box unlearnable as was $\mathcal{F}$.

Specifically, $\mathcal{G}$ will be defined similarly to $\mathcal{F}$, except that its key will also include a seed $\mathbf{s}$ of a pseudo-random function PRF . Instead of mapping the input a to the output b , $g_{k,\mathbf{s}}$ will include two new functions $\text{PRF}_{\mathbf{s}}(x)$ and $\mathbb{G}(x) = b \oplus \text{PRF}_{\mathbf{s}}(x \oplus a)$. Note that without knowing a , an efficient algorithm that gets black-box access to $\text{PRF}_{\mathbf{s}}$ and $\mathbb{G}$ cannot find any correlation between the functions, and therefore, cannot learn b . However, knowing a , we can query $\text{PRF}_{\mathbf{s}}$ with a random string r , query $\mathbb{G}$ with $r \oplus a$, and obtain $b = \text{PRF}_{\mathbf{s}}(r) \oplus \mathbb{G}(r \oplus a)$. Since each individual query to $\mathcal{G}$ is distributed uniformly, we can learn b from a , even if the function errs on, say, a $\frac{1}{4}$ -fraction of inputs. To extract from circuits that make more errors, we use standard direct product amplification [151]; namely, we let $\mathcal{G}$ evaluate $\text{PRF}_{\mathbf{s}}$ and $\mathbb{G}$ on many random inputs in parallel, amplifying the probability of getting the correct answer on a single random input.

In order to successfully extract b , it is not sufficient to deal with circuits that err on a , but we should also handle circuits that do not output b given an encryption of b . We would like to use a similar idea to the one described before; namely, include a function $\mathbb{G}'(\mathbf{c})$ that, on input $\mathbf{c}$, decrypts $\mathbf{c}$ to some plaintext x , and outputs $b \oplus \text{PRF}_{\mathbf{s}}(x \oplus b)$. However, given black-box access to such a function, semantic security is not maintained. Indeed, when two encryptions $\mathbf{c}_1$ and $\mathbf{c}_2$ correspond to the same plaintext then $\mathbb{G}'(\mathbf{c}_1) = \mathbb{G}'(\mathbf{c}_2)$ and when the encryptions correspond to different plaintexts $\mathbb{G}'(\mathbf{c}_1) \neq \mathbb{G}'(\mathbf{c}_2)$ with high probability. Using such an attack it is possible to learn a and thus also b , in a black-box way.

The first step in our solution involves a slight modification to the homomorphic encryption scheme. An encryption of a bit b will include $b \oplus r$ for a random bit r and an encryption of r . We shall assume that encryptions of uniformly random strings are uniformly random themselves. We also modify the decryption and homomorphic evaluation algorithms accordingly. We refer to such an encryption scheme as *decomposable*, since any cipher c is composed of two parts. We think of the encryption of r as a public part and denote it by $\text{pub}(c)$, and of the padded plaintext as a private part. $\text{pub}(c)$ is independent of the plaintext and can be sampled by anyone, but together with the plaintext, it completely determines the cipher.

The next step is to redefine $\mathbb{G}'(c) = b \oplus \text{PRF}_s(x \oplus b || \text{pub}(c))$, where x is the decryption of c . Note that if $c_1 \neq c_2$, and $x_1 = x_2$, then $\text{pub}(c_1) \neq \text{pub}(c_2)$; therefore, also $\mathbb{G}'(c_1)$ and $\mathbb{G}'(c_2)$ are distributed (pseudo) independently at random, and we can show that semantic security is maintained. Additionally, given an encryption of b , it is still possible to learn b by first sampling a random string r and then homomorphically computing an encryption c of $b \oplus r$, and finally, evaluating $\mathbb{G}'(c) \oplus \text{PRF}_s(r || \text{pub}(c)) = b$. Note that both inputs c and $r || \text{pub}(c)$ to $\mathbb{G}'$ and PRF_s are distributed uniformly at random.

Towards the final construction, we still have to bridge two main gaps: first, we would like to rely on one-way functions rather than fully-homomorphic encryption; second, we would like extraction to succeed when the function approximated on random inputs, rather on several specific distributions as described above. In the next two paragraphs, we sketch how we deal with each of the two.

Getting rid of homomorphic encryption. The construction of Barak et al. only uses a standard symmetric encryption, where homomorphic operations are performed gate by gate by f_k : given two encrypted input bits, f_k decrypts them, evaluates the desired operation, and returns the result encrypted. Trying to employ this strategy in our setting we may hope that, since these queries consist of random encryptions, a circuit C that does not err too much will correctly compute the homomorphic operations required for extraction with high enough probability. However, this is not the case since extraction involves a long (a-priori unbounded) homomorphic computation, and even one error along the way could compromise the integrity of the result.

To get around this problem, we use the fact that, in the case of robust unobfuscatable functions, we can assume that C only makes “detectable errors”; we show that this

is enough to recover from errors during the homomorphic computation. Specifically, when a single homomorphic operation fails (the circuit outputs $\perp$), we would like to rerandomize the input encryptions and try again. This could be achieved by relying on rerandomizable encryption. However, we wish to avoid this assumption and provide a solution based on one-way functions. For this purpose, we modify $\mathcal{G}$ to return many random encryptions of the resulting bit, and show a procedure that assures that the extractor is always left with enough “good” encryptions to continue the homomorphic evaluation. (In the actual construction, when dealing with a deterministic $\mathcal{G}$, we will need to use “invoker randomizable” PRFs [16] to make sure that the encryptions are indeed random.)

From several (key-dependent) input distributions to the uniform distribution. As described above, a function $g \in \mathcal{G}$ implements several different functions: $\text{PRF}_s, \mathbb{G}, \mathbb{G}'$, and the homomorphic evaluation functionality. For the extraction procedure described above to succeed, it is not sufficient that C approximates g well on a random input, but rather C should approximate g on each one of these functions individually. Moreover, to guarantee the success of the homomorphic evaluation, we require that C approximates the homomorphic evaluation functionality for *every* possible configuration of gate and input plaintexts. Note that for a specific configuration of gate and input plaintexts, the input to homomorphic evaluation functionality is not distributed uniformly, furthermore, such inputs may not be directly samplable without the secret encryption key. Indeed, in symmetric-key encryption (based on one-way functions), we do not know how to sample encryptions of individual bits without the secret key.

To overcome this issue, we use an encryption scheme that supports sampling ciphers of random plaintexts independently of the secret key. Intuitively, if the circuit does not abort given an encryption of a random bit, with sufficiently high-probability, then it also does not abort for encryptions of specific bits, with some related probability. To make sure that the probability is indeed high-enough we use again parallel repetition. Note that during extraction we must query g with encryptions of individual bits; however, we do not sample those encryptions ourself, but rather these encryptions are obtained as the result of a previous homomorphic operation. Eventually, we show that extraction is successful assuming only that C approximates the resulting function on a uniform

input.

Necessity of one-way functions. We show that one-way functions are not only sufficient, but also necessary for robust unobfuscatable functions. (The function that maps $(k, x_1, \dots, x_m)$ to $(f_k(x_1), \dots, f_k(x_m), x_1, \dots, x_m)$ is one-way for $m \gg |k|$.) This should be contrasted with non-robust unobfuscatable functions, where a similar implication is not known; in fact, Barak et al. [16] construct (non-robust) unobfuscatable Turing machine (rather than circuit) families without any computational assumptions, whereas robust unobfuscatable functions also imply one-way functions in the Turing machine case. Additionally, we do not know what are the minimal assumptions required for verifiable robust unobfuscatable functions.

Constructing verifiable robust unobfuscatable functions. Finally, we show how to “compile” robust unobfuscatable functions into full fledged verifiable robust unobfuscatable functions based on ZAPs (two-message, public-coin witness-indistinguishable protocols) [74] and non-interactive commitments. The high level idea is to put a commitment to the key k in the verification key vk and add to the output of the function a ZAP proving that the function was computed correctly using a key that is consistent with vk . To guarantee that the ZAP does not reveal the secret key k we evaluate two functions in parallel with independent keys and prove that one of them was computed correctly. The construction uses ideas from [58, 78]. Now, given a program that computes the function correctly on some noticeable fraction of the inputs, we can detect when the output is computed incorrectly and replace it with $\perp$.

3.1.7 More Related Work

The resettable-sound protocol constructed in [35] was based on the unobfuscatable functions of [16] and oblivious transfer. Subsequently, Chung, Pass and Seth [63] show how to remove the collision-resistant hashing from the protocols of [15, 69] leading to resettable-sound ZK from one-way functions. The constructions based on robust unobfuscatable function appeared in a subsequent work [36]. The construction of simultaneously-resettable WI arguments from one-way functions [61] was concurrent to [36]. In another concurrent work, Goyal [107] presents a new construction of public-coin concurrent ZK, leading also to a new simultaneously-resettable ZK protocol. The

subsequent work of [60] constructed a four-message resettably-sound ZK protocol from one-way functions.

Chapter Organization

In Section 3.2, we define the notions of robust and verifiable robust unobfuscatable functions, as well as other weaker notions used in our constructions. In Section 3.3, we present our main construction of robust unobfuscatable functions. In Section 3.4, we define verifiable unobfuscatable functions and construct them based on robust unobfuscatable functions. The construction of verifiable unobfuscatable functions will be used to get verifiable robust unobfuscatable functions and applications to ZK. In Section 3.5, we describe our new constructions of resettably-secure protocols. In Section 3.6 we give a transformation from any resettably-sound ZK argument to a family of unobfuscatable functions.

3.2 Robust Unobfuscatable Functions - Definitions

In this section, we define the notion of robust unobfuscatable functions. Before that we define the main notion of function approximation that we work with.

Definition 9 ($(\varepsilon, \nu, \mathcal{D})$ -approximation). *Let $\mathcal{D}$ be a distribution on $\{0, 1\}^n$. A circuit C is said to $(\varepsilon, \nu, \mathcal{D})$ -approximate a function $f : \{0, 1\}^n \rightarrow \{0, 1\}^{n'}$ if:*

1. **It often agrees with f on inputs drawn from $\mathcal{D}$:**

$$\Pr_{q \leftarrow \mathcal{D}} [C(q) = f(q)] \geq \varepsilon ,$$

2. **It almost only makes detectable errors on inputs from $\mathcal{D}$:**

$$\Pr_{q \leftarrow \mathcal{D}} [C(q) \notin \{\perp, f(q)\}] \leq \nu .$$

*C is said to $(\varepsilon, \mathcal{D})$ -approximate f **with errors** if property 2 is not guaranteed.*

Definition 10 (Robust unobfuscatable functions). *A function family $\mathcal{F} = \{f_k\}_{k \in \{0, 1\}^n, n \in \mathbb{N}}$ is a robust unobfuscatable family, with respect to an NP relation $\mathcal{R}_{\mathcal{F}}$ and input sampler $\mathcal{D}$, if it is:*

1. **Black-box unlearnable:** For any poly-size learner L , and any $n \in \mathbb{N}$, L :

$$\Pr_{k \leftarrow \{0,1\}^n} [(k, z) \in \mathcal{R}_{\mathcal{F}} : z \leftarrow L^{f_k}(1^n)] \leq \text{negl}(n) ;$$

namely, L cannot satisfy the relation $\mathcal{R}_{\mathcal{F}}$, given only black-box access to f_k . $\mathcal{R}_{\mathcal{F}}$ is thus called “the unlearnable relation”.

2. **Non-Black-box learnable:** There exists an efficient extractor $\mathcal{E}$ such that, for any functions $\varepsilon = \varepsilon(n) \geq 2^{-\sqrt{n}}$ and $\nu = \nu(n) \leq \frac{\varepsilon}{4n^2}$, any efficient sampler $\mathcal{D} = \mathcal{D}(1^n)$, any large enough $n \in \mathbb{N}$, $k \in \{0,1\}^n$, and circuit C that $(\varepsilon, \nu, \mathcal{D})$ -approximates f_k , $\mathcal{E}$ extracts $z \in \mathcal{R}_{\mathcal{F}}(k)$ from C :

$$\Pr_{\mathcal{E}} [(k, z) \in \mathcal{R}_{\mathcal{F}} : z \leftarrow \mathcal{E}(C, 1^n, 1^{1/\varepsilon})] \geq 1 - \left(\nu^{\Omega(1)} + 2^{-\Omega(n)} \right) \cdot (|C| \cdot n/\varepsilon)^{O(1)} .^1$$

We say that the family is ε -robust for a specific noticeable function $\varepsilon = \varepsilon(n) = n^{-O(1)}$, if there exists an extractor $\mathcal{E}$ such that, for all large enough $n \in \mathbb{N}$, $k \in \{0,1\}^n$, and circuit C that $(\varepsilon, \nu, \mathcal{D})$ -approximates f_k :

$$\Pr_{\mathcal{E}} [(k, z) \in \mathcal{R}_{\mathcal{F}} : z \leftarrow \mathcal{E}(C, 1^n)] \geq 1 - \left(\nu^{\Omega(1)} + 2^{-\Omega(n)} \right) \cdot (|C| \cdot n)^{O(1)} .$$

We next define error-robust unobfuscatable functions, which are defined analogously, only that extraction should work for any circuit C that sufficiently agrees with f_k , even if C makes undetectable errors.

Definition 11 (Error-robust unobfuscatable functions). *A family of functions $\mathcal{F} = \{f_k\}_{k \in \{0,1\}^n, n \in \mathbb{N}}$ is an error-robust unobfuscatable family, with respect to an **NP** relation $\mathcal{R}_{\mathcal{F}}$ and input sampler $\mathcal{D}$, if it is:*

1. **Black-box unlearnable:** As in Definition 10.
2. **Non-Black-box learnable:** There exists an efficient extractor $\mathcal{E}$ such that, for any function $\varepsilon = \varepsilon(n) \geq 2^{-\sqrt{n}}$, any efficient sampler $\mathcal{D} = \mathcal{D}(1^n)$, all large enough $n \in \mathbb{N}$, and any $k \in \{0,1\}^n$, $\mathcal{E}$ extracts $z \in \mathcal{R}_{\mathcal{F}}(k)$ from any circuit C that

¹In most cases throughout this chapter, the extraction error expression above can be simplified to $\text{negl}(n)$, e.g., when $|C| = \text{poly}(n)$, $\varepsilon = 1/\text{poly}(n)$, and $\nu = \text{negl}(n)$. There are though cases in which we may use the more general bound above, e.g., in Section 3.5.4.2.

$(\frac{1}{2} + \varepsilon, \mathcal{D})$ -approximates f_k **with errors**:

$$\Pr_{\mathcal{E}} \left[(k, z) \in \mathcal{R}_{\mathcal{F}} : z \leftarrow \mathcal{E}(C, 1^n, 1^{1/\varepsilon}) \right] \geq 1 - 2^{-\Omega(n)} \cdot |C|^{O(1)} .$$

Remark 12 (The sampler $\mathcal{D}$). *In the above definitions, we allow the input sampler $\mathcal{D}$ to represent an arbitrary samplable distribution, where sampling is independent of the key k for the unobfuscatable function. We can consider a more strict (but natural) definition, where $\mathcal{D}(1^n)$ represents the uniform distribution over strings in $\{0, 1\}^n$. Indeed, our constructions also achieve this variant (see Remark 28).*

Remark 13 (Uniquely determined functions). *For the case of erroneous circuits, i.e. with non-detectable errors (Definition 11), we require for simplicity that the circuit C determines one specific function, and hence we require more than $\frac{1}{2}$ -agreement (this is necessary to guarantee that C cannot simultaneously approximate two functions in the family, and also sufficient assuming that any two functions in the family have no agreement.) In principle, one may also consider alternative definitions of error-robustness where the circuit may somewhat agree with several functions and the extractor is required to extract the keys of all the functions whose agreement with C crosses a given threshold. We restrict attention to the unique-function case, although our techniques naturally extends to such alternative notions.*

Remark 14 (unique-witness unlearnable relations). *A natural requirement regarding the unlearnable relation $\mathcal{R}_{\mathcal{F}}$ of a family $\mathcal{F}$ is that there exists a unique witness $z \in \mathcal{R}_{\mathcal{F}}(k)$; namely, z is an efficiently computable deterministic function of k . Our constructions satisfy this unique-witness requirement.*

We also note that this property allows to rule out the same notion of virtual black-box obfuscation ruled out in [16] that prevents the adversary from learning any deterministic function of the key k .

Remark 15 (Extraction from computationally-bounded circuits). *As stated above extraction in Definitions 10 and 11 is information theoretic, and does not explicitly require that the approximating circuit C is of poly-size (although this is the typical case in applications).*

Here there is a difference between Definition 10 and Definition 11. In Definition 10, the fraction ν of undetectable errors is explicit in the definition and affects the probability that extraction will indeed succeed; in particular, we do not require that ν is a

concrete negligible function. Jumping ahead, when extracting from poly-size circuits, we may prevent undetectable errors using, for example, a computationally-sound proof system, in which case the function ν may be some negligible function $\text{negl}(n)$ that depends on the concrete poly-size family of circuits we are extracting from. In Definition 11, we require that extraction works even with undetectable errors, and as said we do not a-priori restrict the circuits to poly-size. To achieve such extraction, we will indeed rely on statistically-sound proofs systems (concretely ZAPs). It is possible to augment the definition to only consider poly-size circuit families, in which case there will be some negligible error $\text{negl}(n)$ in the extraction probability, which depends on the concrete family of circuits.

Remark 16 (Comparison with strongly unobfuscatable functions of [16]). Barak et al. define and construct a family of functions that is strongly unobfuscatable. Such functions satisfy a weaker extraction requirement than error-robust unobfuscatable functions, and therefore only rule out a stronger definition of approximate obfuscation.

The definition of [16] considers extraction from a circuit that is sampled from a distribution $\mathcal{C}$ over circuits (this corresponds to the output distribution of an obfuscator on a specific input program with random coins). The extraction requirement states that if, for every input q a circuit $C \leftarrow \mathcal{C}$ satisfies $C(q) = f_k(q)$ with overwhelming probability over the choice of C , then the extractor on input circuit $C \leftarrow \mathcal{C}$ outputs k with overwhelming probability. In particular, extraction requires that, for every input, the circuit is correct with overwhelming probability. In contrast, the extraction requirement of error-robust unobfuscatable functions is guaranteed even if for some inputs the circuit never answers correctly (in fact this could be the case for almost all inputs).

3.2.1 Robustness from Weaker Robustness

In this section, we discuss two natural relaxations of robust obfuscation that will be convenient to work with, and show how to transform functions that are unobfuscatable according to these notions to functions that are robust according to Definition 10. We restrict attention to approximation **without errors**. (Eventually, in Section 3.4, we show how to go from robust functions to error-robust ones.)

First, we show that, using standard direct product amplification [151], we can always boost the robustness of an unobfuscatable family (at the cost of blowing up the input and output size).

Lemma 17. Any $\left(1 - \frac{1}{\sqrt[4]{n}}\right)$ -robust family $\mathcal{G}$ can be transformed to a robust family $\mathcal{F}$, where the input and output size grow by a n factor. Furthermore, if the extractor of $\mathcal{G}$ runs in time $\text{poly}(n) \cdot |C|^d$, where C is the approximation circuit for g_k , then the extractor of $\mathcal{F}$ runs in time $\text{poly}(n) \cdot (|C|/\varepsilon)^d$, where C is a $(\varepsilon, \nu, \mathcal{D})$ -approximation circuit for f_k .

We describe the construction behind the lemma, followed by the analysis.

Construction 3.2.1 (Amplifying robustness). Let $\mathcal{G}$ be a $\left(1 - \frac{1}{\sqrt[4]{n}}\right)$ -robust unobfuscatable family, with respect to an unlearnable relation $\mathcal{R}_{\mathcal{F}}$, and an input distribution ensemble $\mathcal{D}$; We construct a new robust unobfuscatable family $\mathcal{F}$, with respect to the same relation $\mathcal{R}_{\mathcal{F}}$, and the n -fold distribution ensemble $\widehat{\mathcal{D}} = \mathcal{D} \times \dots \times \mathcal{D}$. Each function $f_k : \{0, 1\}^{n^2} \rightarrow \{0, 1\}^{n^2}$ is just the n -fold version $\widehat{g}_k(q_1, \dots, q_n) = (g_k(q_1), \dots, g_k(q_n))$ of the function $g_k : \{0, 1\}^n \rightarrow \{0, 1\}^n$.

Proof of Lemma 17. Unlearnability of $\mathcal{F}$ follows directly from that of $\mathcal{G}$, since any oracle call to $f_k \in \mathcal{F}$ can be simulated by n oracle calls to the corresponding g_k .

To show that the function is robust, we describe its extractor $\widehat{\mathcal{E}}$. For functions $\varepsilon(n) \geq 2^{-\sqrt{n}}$, $\nu(n) \leq \frac{\varepsilon}{4n^2}$ the extractor $\widehat{\mathcal{E}}$ is given as input a circuit C that $(\varepsilon, \nu, \widehat{\mathcal{D}})$ -approximates f_k , as well as the parameter $1/\varepsilon$. $\widehat{\mathcal{E}}$ first constructs from C a probabilistic circuit C' that, with very high probability, will give a very good approximation of g_k .

The circuit C' , given a random sample $q \leftarrow \mathcal{D}(1^n)$, samples $(q_1, \dots, q_n) \leftarrow \widehat{\mathcal{D}}(1^n)$, replaces a random coordinate q_i with q , and feeds the augmented n -fold sample to C . The circuit C' repeats this procedure $\frac{n^2}{\varepsilon}$ times (for the same q), and if it obtains an answer $(a_1, \dots, a_n)$ that is not $\perp$, it returns a_i . Then, after producing the circuit C' , $\widehat{\mathcal{E}}$ chooses n random strings $r_1, \dots, r_n$ for C' , and runs the extractor $\mathcal{E}$ of g_k , on each $C'(\cdot; r_i)$. If all executions output $\perp$, the extractor fails, and otherwise it produces k .

To show that $\widehat{\mathcal{E}}$ works as required, we show that the circuit C' is likely to approximate g_k sufficiently well. For any sample $q \in \text{supp}(\mathcal{D})$, let $\widehat{\mathcal{D}}|_q$ denote the distribution that the circuit C' samples from (i.e., where a random sample is drawn from $\widehat{\mathcal{D}}$, and then one of its random coordinates is replaced by q). We say that q is **good** if $\Pr_{\vec{q} \leftarrow \widehat{\mathcal{D}}|_q} [C(\vec{q}) \neq \perp] \geq \varepsilon/2n$.

We claim that a sample q drawn from $\mathcal{D}$ is **good** with probability at least $1 - \frac{\log(2/\varepsilon)}{n} \geq 1 - O\left(\frac{1}{\sqrt{n}}\right)$. Indeed, if that was not the case, then in a sample $\vec{q} \leftarrow \widehat{\mathcal{D}}(1^n)$, there is

some coordinate that is **not good**, except with probability $\varepsilon/2$; more precisely,

$$\Pr_{\vec{q} \leftarrow \widehat{\mathcal{D}}} [q_1, \dots, q_n \in \text{good}] \leq \prod_{i \in [n]} \Pr_{q_i \leftarrow \mathcal{D}} [q_i \in \text{good}] \leq \left(1 - \frac{\log(2/\varepsilon)}{n}\right)^n \leq \varepsilon/2 .$$

However, the probability that $C(\vec{q}) \neq \perp$ and some q_i is not **good** is at most $\varepsilon/2$; indeed,

$$\begin{aligned} \frac{1}{n} \cdot \Pr_{\vec{q} \leftarrow \widehat{\mathcal{D}}} [C(\vec{q}) \neq \perp, \exists i : q_i \notin \text{good}] &\leq \\ \Pr_{\vec{q} \leftarrow \widehat{\mathcal{D}}, i \leftarrow [n]} [C(\vec{q}) \neq \perp, q_i \notin \text{good}] &\leq \\ \Pr_{q \leftarrow \mathcal{D}, \vec{q} \leftarrow \widehat{\mathcal{D}}|_q} [C(\vec{q}) \neq \perp, q \notin \text{good}] &\leq \\ \Pr_{q \leftarrow \mathcal{D}|\overline{\text{good}}, \vec{q} \leftarrow \widehat{\mathcal{D}}|_q} [C(\vec{q}) \neq \perp] &\leq \\ \varepsilon/2n , \end{aligned}$$

where $\mathcal{D}|\overline{\text{good}}$ denotes $\mathcal{D}$ conditioned on not hitting the set **good**. Overall $C(\vec{q}) \neq \perp$ with probability less than $\varepsilon/2 + \varepsilon/2 \leq \varepsilon$, leading to a contradiction.

Next, note that, conditioned on q being **good**, $C'(q) \neq \perp$, except with probability $(1 - \frac{\varepsilon}{2n})^{n^2/\varepsilon} \leq 2^{-n/2}$, and thus

$$\Pr_{q \leftarrow \mathcal{D}, r} [C'(q; r) \neq \perp] \geq \left(1 - O\left(\frac{1}{\sqrt{n}}\right)\right) \cdot (1 - 2^{-\Omega(n)}) \geq \left(1 - O\left(\frac{1}{\sqrt{n}}\right)\right) .$$

In addition, since $C(\vec{q}) \notin \{\perp, \widehat{g}_k(\vec{q})\}$ with probability at most $\nu(n)$, by a union bound

$$\Pr_{q \leftarrow \mathcal{D}, r} [C'(q; r) \notin \{\perp, g_k(q)\}] \leq \nu(n) \cdot \frac{n^2}{\varepsilon} \leq \frac{1}{4} .$$

Combining these two facts, it follows that for a $\left(1 - \frac{1}{\sqrt[4]{n}} - \frac{1}{2}\right)$ -fraction of the random coins r used by C' , it holds that both

$$\begin{aligned} \Pr_{q \leftarrow \mathcal{D}} [C'(q; r) \neq \perp] &\geq 1 - O\left(\frac{1}{\sqrt[4]{n}}\right) \\ \Pr_{q \leftarrow \mathcal{D}} [C'(q; r) \notin \{\perp, \widehat{g}_k(\vec{q})\}] &\geq 1 - \sqrt{\nu(n) \cdot \frac{n^2}{\varepsilon}} ; \end{aligned}$$

namely, $C'(\cdot; r)$ is a $\left(1 - O\left(\frac{1}{\sqrt[4]{n}}\right), \sqrt{\nu(n) \cdot \frac{n^2}{\varepsilon}}, \mathcal{D}\right)$ -approximation of g_k . For such ran-

domness r , we know that the extractor $\mathcal{E}_{\mathcal{G}}$ for g_k outputs k with probability at least

$$1 - \left(\sqrt{\nu(n) \cdot \frac{n}{\varepsilon}} \right)^{\Omega(1)} \cdot (|C'| \cdot n)^{O(1)} = 1 - \nu^{\Omega(1)} (|C| \cdot n/\varepsilon)^{O(1)} .$$

and since there are n trials, the overall extraction probability is

$$1 - \nu^{\Omega(1)} (|C| \cdot n/\varepsilon)^{O(1)} - \left(1 - \frac{1}{2} - \frac{1}{\sqrt[4]{n}} \right)^n \geq 1 - \left(\nu^{\Omega(1)} + 2^{-\Omega(n)} \right) \cdot (|C| \cdot n/\varepsilon)^{O(1)} ,$$

as required.

Noting that the size of C' is $\text{poly}(n) \cdot |C|/\varepsilon$, and that we make n trials to extract from C' ; we deduce that, if $\mathcal{E}_{\mathcal{G}}$ runs in time $\text{poly}(n) \cdot |C|^d$, then the extractor $\mathcal{E}_{\mathcal{F}}$ runs in time $\text{poly}(n) \cdot (|C|/\varepsilon)^d$. $\square$

When constructing robust unobfuscatable functions, it will be convenient to also consider a further relaxed notion where, in order to extract, the circuit is required to approximate the function on multiple distributions $\mathcal{D}_1, \dots, \mathcal{D}_d$, instead of one. Throughout, it will always be the case that d is a constant. This definition will only serve us as an intermediate step towards constructing families according to Definition 10, where $d = 1$. As another relaxation, we will not require that each of these distributions can be sampled independently of k , but rather we will only require that, when i^* is chosen at random from $[d]$, it is possible to sample from $\mathcal{D}_{i^*}$, independently of k . As we shall see later on, the latter weak sampling guarantee (combined with parallel repetition) will be sufficiently powerful to construct an unobfuscatable function with a single distribution that is samplable independently of k .

We next present the definition. Since robustness can always be amplified, it will be sufficient and more convenient to describe the notion for some concrete constant approximation factor $1 - \delta$ (we will eventually set $\delta < 2^{-5}$, see Section 3.3).

Definition 18 (*d -distribution robust unobfuscatable function*). *A family of functions $\mathcal{F}$ is a d -distribution robust unobfuscatable family with respect to an NP relation $\mathcal{R}_{\mathcal{F}}$ and input samplers $\mathcal{D}_1, \dots, \mathcal{D}_d$, if it is:*

1. **Black-box unlearnable:** *as in Definition 10.*
2. **Non-Black-box learnable:** *There exists an efficient extractor $\mathcal{E}$ such that, for any function $\nu = \nu(n)$, for any large enough $n \in \mathbb{N}$, any $k \in \{0, 1\}^n$, and every*

circuit C that $(1 - \delta, \nu, \mathcal{D}_i)$ -approximates f_k , for all $i \in [d]$, $\mathcal{E}$ extracts $z \in \mathcal{R}_{\mathcal{F}}(k)$ from C :

$$\Pr_{\mathcal{E}}[(k, z) \in \mathcal{R}_{\mathcal{F}} : z \leftarrow \mathcal{E}(C, 1^n)] \geq 1 - \nu^{\Omega(1)} \cdot (|C| \cdot n)^{O(1)},$$

where $\mathcal{D}_i = \mathcal{D}_i(1^n, k)$.

Definition 19 (jointly key-independent distributions). *For any $k \in \{0, 1\}^n$, consider the distribution*

$$\mathcal{D}_k^* = \{q \leftarrow \mathcal{D}_{i^*}(1^n, k) \mid i^* \leftarrow [d]\}$$

We say that $\mathcal{D}_1, \dots, \mathcal{D}_d$ are **jointly key-independent**, if there exists an efficient sampler $\mathcal{D}^*$ such that for all k , $\mathcal{D}^*(1^n)$ generates samples from the distribution $\mathcal{D}_k^*$.

We show that any d -distribution robust unobfuscatable family, where $\mathcal{D}_1, \dots, \mathcal{D}_d$ are jointly key-independent, can be transformed into a robust unobfuscatable family according to Definition 10. Specifically, we show a $(1 - \frac{\delta}{d})$ -robust family, and then robustness can be amplified using Lemma 17.

Lemma 20. *If $\mathcal{G}$ is a d -distribution robust unobfuscatable family, with samplers $\mathcal{D}_1, \dots, \mathcal{D}_d$ that are jointly key-independent, then $\mathcal{G}$ is also $(1 - \frac{\delta}{d})$ -robust with respect to the (key-independent) distribution $\mathcal{D}^* = \mathcal{D}^*(1^n)$.*

Proof. Indeed, by a simple union bound, any circuit that $(1 - \frac{\delta}{d}, \nu, \mathcal{D}^*)$ -approximates f_k must $(1 - \delta, \nu, \mathcal{D}_i)$ -approximate f_k for each one of the distributions $\mathcal{D}_1, \dots, \mathcal{D}_d$. Thus, we can use the existing extractor associated with $\mathcal{G}$. $\square$

3.2.2 Robust Unobfuscatable Functions with a Hardcore Secret

The notion of robust (and error-robust) unobfuscatable functions is defined with respect to an unlearnable relation $\mathcal{R}_{\mathcal{F}}$, where it is guaranteed that, given black-box access to f_k , it is hard to fully learn some unlearnable secret $z \in \mathcal{R}_{\mathcal{F}}(k)$. We now present a natural variant of this definition, where each function has an associated hardcore secret that is indistinguishable from a random string, even given black-box access to the function; in contrast, the hardcore secret can be extracted from any circuit that approximates f_k . This variant will be useful for constructing unobfuscatable functions with a verifiable

unlearnable secret (see Section 3.4), and for some applications to resettable protocols (see Section 3.5.6).

In the following definition, $\mathcal{HC}_n = \{h : \{0, 1\}^{\text{poly}(n)} \rightarrow \{0, 1\}^n\}$ will represent a family of functions (called hardcore functions).

Definition 21 (Robust unobfuscatable functions with a hardcore secret). *A family of functions $\mathcal{F} = \{f_k\}_{k \in \{0, 1\}^n, n \in \mathbb{N}}$ is a robust unobfuscatable family with respect to an efficient input sampler $\mathcal{D}$, and a family of **hardcore functions** $\mathcal{HC}$, if it has the following properties:*

1. **Black-box indistinguishability:** *For any poly-size distinguisher L , and any $n \in \mathbb{N}$:*

$$\left| \Pr_{(k, h)} [L^{f_k}(h, h(k)) = 1] - \Pr_{(k, h, u)} [L^{f_k}(h, u) = 1] \right| \leq \text{negl}(n) ,$$

where $k \leftarrow \{0, 1\}^n$, $h \leftarrow \mathcal{HC}_n$, and $u \leftarrow \{0, 1\}^n$ are all sampled independently.

2. **Non-Black-box learnability:** *There exists an efficient extractor $\mathcal{E}$ such that, for any functions $\varepsilon = \varepsilon(n) \geq 2^{-\sqrt{n}}$, $\nu = \nu(n) \leq \frac{\varepsilon}{4n^2}$, any efficient sampler $\mathcal{D} = \mathcal{D}(1^n)$, any large enough n , any hardcore function $h \in \mathcal{HC}_n$, any $k \in \{0, 1\}^n$, and every circuit C that $(\varepsilon, \nu, \mathcal{D})$ -approximates f_k , $\mathcal{E}$ extracts $h(k)$ from C :*

$$\Pr_{\mathcal{E}} [h(k) \leftarrow \mathcal{E}(C, h, 1^n, 1^{1/\varepsilon})] \geq 1 - \left(\nu^{\Omega(1)} + 2^{-\Omega(n)} \right) \cdot (|C| \cdot n / \varepsilon)^{O(1)} .$$

We show that any robust unobfuscatable family $\mathcal{G}$ that has a unique-witness unlearnable relation $\mathcal{R}_{\mathcal{G}}$ (as defined in Remark 14) can be transformed into a robust $\mathcal{F}$ with a hardcore secret.

Lemma 22. *There exists a hardcore family $\mathcal{HC} = \{\mathcal{HC}_n\}_{n \in \mathbb{N}}$ such that a robust unobfuscatable family $\mathcal{G}$ with respect to a unique-witness unlearnable relation $\mathcal{R}_{\mathcal{G}}$, can be transformed to a robust unobfuscatable family $\mathcal{F}$ with respect to some relation $\mathcal{R}_{\mathcal{F}}$ and the hardcore family $\mathcal{HC}$.*

Proof. Given a robust family $\mathcal{G}$, with respect to a unique-witness unlearnable relation $\mathcal{R}_{\mathcal{G}}$ and an efficient input sampler $\mathcal{D}$, we define a new function family $\mathcal{F}$, where each f_k consists of n independently chosen functions from $\mathcal{G}$, with keys $(k'_1, \dots, k'_n)$; the new

key k is set to be $(k'_1, \dots, k'_n)$. The input sampler of the function is the n -fold product distribution $\widehat{\mathcal{D}} = \mathcal{D} \times \dots \times \mathcal{D}$, and $f_k(q_1, \dots, q_n)$ is defined to be $g_{k'_1}(q_1), \dots, g_{k'_n}(q_n)$.

Let us denote by $\ell = \ell(n)$ the length $\ell = |z_i|$ of the unique-witness $z_i \in \mathcal{R}_{\mathcal{G}}(k'_i)$. The hardcore family $\mathcal{HC}_n$ will be the family $\mathcal{GL}_n$ that extracts a single Goldreich-Levin [92] hardcore-bit from each z_i . That is, a randomly chosen function $h_{r_1, \dots, r_n} \in \mathcal{GL}_n$ is parameterized by n random strings $r_1, \dots, r_n \in \{0, 1\}^\ell$, and is defined as:

$$h_{r_1, \dots, r_n}(k'_1, \dots, k'_n) = \langle z_1, r_1 \rangle, \dots, \langle z_n, r_n \rangle ,$$

where each $z_i \in \mathcal{R}_{\mathcal{G}}(k'_i)$ is the unique unlearnable secret corresponding to k'_i , and $\langle \cdot, \cdot \rangle$ is the inner-product operation modulo 2.

To see that $\mathcal{F}$ has the black-box indistinguishability property (Definition 21), note that each secrets z_i is unlearnable given oracle access to $g_{k'_i}$. We can thus apply the Goldreich-Levin theorem, as stated in Corollary 6, to deduce that $\langle z_i, r_i \rangle$ is pseudo-random given r_i and oracle access to $g_{k'_i}$. Thus, by a standard hybrid argument $\langle z_1, r_1 \rangle, \dots, \langle z_n, r_n \rangle$ are pseudo-random, given $r_1, \dots, r_n$ and oracle access to f_k .

Next, we show that $\mathcal{F}$ is non-black-box learnable according to Definition 21; namely, the hardcore secret can be extracted from any circuit approximation. Indeed, note that any circuit C that $(\varepsilon, \nu, \widehat{\mathcal{D}})$ -approximates f_k can be transformed, with overwhelming probability, into a circuit that $(\varepsilon/2, \sqrt{\nu}, \mathcal{D})$ -approximates $g_{k'_i}$ (and this holds for any $i \in [n]$). To transform C into such a circuit, we construct a circuit C' that, given a sample $q \leftarrow \mathcal{D}(1^n)$ for $g_{k'_i}$, completes q into a sample from $\widehat{\mathcal{D}}$ by sampling the rest of the coordinates himself, and then feeds this tuple to C . A standard averaging argument shows that with probability at least $\varepsilon/2 - \sqrt{\nu} \geq \varepsilon/4$ over the choice of randomness r for C' the resulting circuit $C'(\cdot; r)$ $(\varepsilon/2, \sqrt{\nu}, \mathcal{D})$ -approximates $g_{k'_i}$. Thus we can take n/ε random copies of C' , and with probability $1 - 2^{-n}$ ensure that one of the circuits $(\varepsilon/2, \sqrt{\nu}, \mathcal{D})$ -approximates $g_{k'_i}$.

Thus the extractor $\mathcal{EF}$ for $\mathcal{F}$, would run the extractor $\mathcal{EG}$ with each of the n instances of circuit C' , and obtain the corresponding secret $z_i \in \mathcal{R}_{\mathcal{G}}(k'_i)$, with probability $1 - \nu^{\Omega(1)} (|C| \cdot n/\varepsilon)^{O(1)}$. In particular, given $h_{r_1, \dots, r_n} \in \mathcal{GL}_n$, $\mathcal{EG}$ can compute as required

$$h_{r_1, \dots, r_n}(k) = h_{r_1, \dots, r_n}(k'_1, \dots, k'_n) = \langle z_1, r_1 \rangle, \dots, \langle z_n, r_n \rangle .$$

This concludes the proof of Lemma 22. $\square$

3.3 A Construction of Robust Unobfuscatable Functions

In this section, we construct robust unobfuscatable functions from one-way functions.

Theorem 23. *Assuming one-way functions, there exists a family of robust unobfuscatable functions.*

3.3.1 Required PRFs and Encryption

Before describing the construction, we define several required primitives.

Invoker randomizable pseudo-random functions Invoker randomizable pseudo-random functions allow their invoker to ensure that the output is truly uniform, independently of the seed for the function. Looking ahead, such functions will allow the extractor for the unobfuscatable function family to obtain samples from the proper distribution. See further details in the next section. The definition is taken almost verbatim from [16].

Definition 24 (Invoker randomizable pseudo-random functions [16]). *Let $\text{PRF} = \{\text{PRF}_s\}_{s \in \{0,1\}^*}$, be pseudo random function family, where for $s \in \{0,1\}^n$, $\text{PRF}_s : \{0,1\}^{\ell(n)+n} \rightarrow \{0,1\}^n$. Then PRF is called **invoker randomizable** if, for any $s \in \{0,1\}^n$ and any $x \in \{0,1\}^{\ell(n)}$, the mapping $r \mapsto \text{PRF}_s(x, r)$ is a permutation.*

Barak et al. [16] construct invoker randomizable PRFs from any PRF.

Decomposable CCA-1 symmetric-key encryption In our construction, we will make use of *symmetric-key encryption* schemes with specific properties. We next define these properties and note existing constructions satisfying them, based only on one-way functions.

In a nutshell, we require a CCA-1 symmetric-key encryption scheme with public randomness, and oblivious generation of ciphers for random plaintexts. Such an encryption scheme can be obtained using a one-bit output PRF by encrypting $\text{Enc}_{\text{sk}}(m; r) = (r, m \oplus \text{PRF}_{\text{sk}}(r))$. In what follows, we give slightly more general definitions that are sufficient for our needs, and will be useful for optimization of extraction running time.

First, we require that the encryption scheme is *decomposable*, meaning that every ciphertext can be “decomposed” into a *public part* and a *private part*. We require that

the public part is independent of the plaintext, however, together with the plaintext, the public part uniquely defines the ciphertext (with respect to a given secret key). For example, any encryption that uses public randomness (i.e., the randomness of the encryption algorithm is included in the ciphertext) is decomposable; the public part of the cipher is just the public randomness.

Definition 25 (Decomposable encryption). *An encryption scheme $(\text{Gen}, \text{Enc}, \text{Dec})$ is decomposable if there exist an efficient algorithm pub that operates on ciphertexts and satisfies the following conditions:*

- *For ciphertext c , $\text{pub}(c)$ is independent of the plaintext and samplable; that is, there exist an efficient sampler PubSamp such that, for any secret key $\text{sk} \in \{0, 1\}^n$:*

$$\text{PubSamp}(1^n) \triangleq \text{pub}(\text{Enc}_{\text{sk}}(0)) \triangleq \text{pub}(\text{Enc}_{\text{sk}}(1)) .$$

- *A ciphertext c is deterministically defined by $\text{pub}(c)$ and the plaintext; that is, for every secret key sk and two distinct ciphers $c \neq c'$, if $\text{pub}(c) = \text{pub}(c')$, then $\text{Dec}_{\text{sk}}(c) \neq \text{Dec}_{\text{sk}}(c')$.*

In addition, we shall require random generation of ciphertexts encrypting random values.

Definition 26 (Encryption with random ciphertext generation). *An encryption scheme $(\text{Gen}, \text{Enc}, \text{Dec})$ is said to have random ciphertext generation if there exist a ciphertext sampling algorithm RanSamp such that for any secret key $\text{sk} \in \{0, 1\}^n$:*

$$\text{RanSamp}(1^n) \triangleq \text{Enc}_{\text{sk}}(U_1) ,$$

where U_1 is the uniform distribution over $\{0, 1\}$.

We remark that in both Definitions 25 and 26, the equality of distributions can be naturally replaced with statistical (or computational) indistinguishability; however, the construction presented below does satisfy the stronger notion with equality. We also remark that if we assume that the scheme has random ciphertext generation, then the algorithm PubSamp , can be simply implemented as $\text{PubSamp}(1^n) \triangleq \text{pub}(\text{RanSamp}(1^n))$, and need not be explicitly defined.

The encryption scheme used in our constructions We will use a CCA-1 symmetric key decomposable bit encryption scheme with random ciphertext generation. Such an encryption scheme can be constructed from one-way functions. Concretely, given a PRF $\{f_s\}_{s \in \{0,1\}^*}$ with one bit output, for security parameter n , the secret key is a random $s \in \{0,1\}^n$, and encryption of a bit $m \in \{0,1\}$ is computed by sampling a random $r \in \{0,1\}^n$ and outputting $r, f_s(r) \oplus m$. This function can be shown to be CCA-1 (see [93]), it is clearly decomposable and has random cipher generation.

3.3.2 The Construction

To prove Theorem 23, we construct an unobfuscatable function family that is d -distribution robust with respect to distribution ensembles $(\mathcal{D}_1, \dots, \mathcal{D}_d)$ that are jointly key-independent in the sense of Definition 19. Then, by Lemmas 20 and 17 we can obtain robust unobfuscatable functions according to Definition 10.

Our construction follows the high-level overview provided in Section 3.1.6, we now proceed to describe it in detail.

Construction 3.3.1. *Let (a, b) be n -bit strings, sk a key for a CCA-1 symmetric key for a decomposable encryption scheme with random ciphertext generation (Definitions 25, 26), and let s be seed for a pseudo-random function PRF mapping $\{0,1\}^*$ to $\{0,1\}^n$. The construction will use two probabilistic and three deterministic auxiliary functions:*

1. *The function $\mathbb{A}_{sk,a}$, given any input, returns a bit encryption $c = \text{Enc}_{sk}(a)$ of a 's bits.*
2. *The function $\mathbb{H}_{sk}$ is given two encryptions (c_1, c_2) , and an operation $\odot$ in a universal set of gates; it then decrypts to obtain the plaintexts x_1 and x_2 , where $x_i = \text{Dec}_{sk}(c_i)$, it computes the operation: $x_3 = x_1 \odot x_2$, and returns an encryption $c_3 = \text{Enc}_{sk}(x_3)$.*
3. *The function $\mathbb{R}_s$ is given $r \in \{0,1\}^*$, and returns $\text{PRF}_s(r)$.*
4. *The function $\mathbb{R}_{a,b,s}$ is given $r \in \{0,1\}^n$, and returns $b \oplus \text{PRF}_s(r \oplus a)$.*
5. *The function $\mathbb{R}_{sk,b,s}$ is given n bit encryptions $(c_1, \dots, c_n)$, it decrypts them to obtain a plain text $r \in \{0,1\}^n$, and returns $b \oplus \text{PRF}_s(r \oplus b || \text{pub}(c_1) || \dots || \text{pub}(c_n))$, where $\text{pub}(c)$ is the public part of c .*

Making the functions deterministic and invoker randomizable Let s' be a seed for an invoker randomizable PRF. We define derandomized variants $\mathbb{A}_{\text{sk},a,s'}, \mathbb{H}_{\text{sk},s'}$ of the two probabilistic functions: $\mathbb{A}_{\text{sk},a}, \mathbb{H}_{\text{sk}}$. For a function $\mathbb{G}_{\text{key}} \in \{\mathbb{H}_{\text{sk}}, \mathbb{A}_{\text{sk},a}\}$, the function $\mathbb{G}_{\text{key},s'}$ gets, in addition to its original input q , a random string $r \in \{0, 1\}^{\text{poly}(n)}$. $\mathbb{G}_{\text{key},s'}(q; r)$ runs the original $\mathbb{G}_{\text{key}}(q)$, using randomness $\text{PRF}_{s'}(q; r)$.

The n -fold functions $\widehat{\mathbb{A}}_{\text{sk},a,s'}, \widehat{\mathbb{H}}_{\text{sk},s'}$ We define variants of $\mathbb{A}_{\text{sk},a,s'}, \mathbb{H}_{\text{sk},s'}$, that instead of returning a single cipher, return an n -fold cipher, consisting of n independent ciphers encrypting the same result. Specifically, for $\mathbb{G} \in \{\mathbb{A}_{\text{sk},a,s'}, \mathbb{H}_{\text{sk},s'}\}$, $\widehat{\mathbb{G}}$ gets input $(q, r_1, \dots, r_n)$ and returns $\mathbb{G}(q, r_1), \dots, \mathbb{G}(q, r_n)$.

The actual function A function $f_k \in \mathcal{F}$ will be parameterized by a random $k = (a, b, \text{sk}, s, s') \in \{0, 1\}^{5n}$, where (a, b, sk, s, s') are as specified above. The function f_k will be given input $(q, i_{\mathbb{G}})$ where $i_{\mathbb{G}} \in [5]$, indicates which function $\mathbb{G} \in \{\widehat{\mathbb{A}}_{\text{sk},a,s'}, \widehat{\mathbb{H}}_{\text{sk},s'}, \mathbb{R}_s, \mathbb{R}_{a,b,s}, \mathbb{R}_{\text{sk},b,s}\}$ to invoke with input q .

The unlearnable relation is

$$\mathcal{R}_{\mathcal{F}} = \left\{ ((a, b, \text{sk}, s, s'), \tilde{b}) : b = \tilde{b} \right\}.$$

(Note that this is a unique-witness relation in the sense of Remark 14.)

3.3.3 Black-Box Unlearnability

We now prove black-box unlearnability. The high-level overview of the proof is presented in Section 3.1.6.

Lemma 27. $\mathcal{F}$ given by Construction 3.3.1 satisfies black-box unlearnability with respect to $\mathcal{R}_{\mathcal{F}}$.

Proof. We shall perform the analysis in several steps: we will first show unlearnability assuming probabilistic oracles, where $\mathbb{A}_{\text{sk},a}, \mathbb{H}_{\text{sk}}$ are given in their 1-fold version; then, we will show security for the case that randomness is derived using PRFs; finally, we will move to security for n -fold $\widehat{\mathbb{A}}_{\text{sk},a}, \widehat{\mathbb{H}}_{\text{sk}}$, and for the actual function f_k .

Let $\mathcal{A}$ be a polysize adversary with oracle access to the functions $\mathbb{H}_{\text{sk}}, \mathbb{R}_s, \mathbb{R}_{\text{sk},b,s}$. Let E_1 be the event that $\mathcal{A}$ produces distinct queries $q = (c_1, \dots, c_n), q' = (c'_1, \dots, c'_n)$

to $\mathbb{R}_{\text{sk},b,s}$ such that

$$r \oplus b || \text{pub}(\mathbf{c}_1) || \dots || \text{pub}(\mathbf{c}_n) = r' \oplus b || \text{pub}(\mathbf{c}'_1) || \dots || \text{pub}(\mathbf{c}'_n) ,$$

where $(r, r') \in \{0, 1\}^n$ are the decryptions under sk of (q, q') .

Claim 3.3.1. $\Pr_{b,\text{sk},s}[E_1] = 0$.

Proof. Indeed, for E_1 to occur, it must be that, for all $i \in [n]$, $r_i = r'_i$ and $\text{pub}(\mathbf{c}_i) = \text{pub}(\mathbf{c}'_i)$, implying that $\mathbf{c}_i = \mathbf{c}'_i$, and thus also $q = q'$; indeed, recall that, in decomposable encryption (Definition 25), the public part and the plaintext determine the cipher. $\square$

We now define E_2 to be the event that $\mathcal{A}$ produces queries $q = (\mathbf{c}_1, \dots, \mathbf{c}_n)$ to $\mathbb{R}_{\text{sk},b,s}$ and q' to $\mathbb{R}_s$ such that $r \oplus b || \text{pub}(\mathbf{c}_1) || \dots || \text{pub}(\mathbf{c}_n) = q'$, where as above $r \in \{0, 1\}^n$ is the underlying plaintext of q .

Claim 3.3.2. $\Pr_{b,\text{sk},s}[E_2] \leq \text{negl}(n)$.

Proof. First, we note that the event is testable given (b, sk) and thus, it is enough to show that the claim holds when PRF_s is replaced by a truly random function R .

Let $\neg E^i$ be the event that E_2 does **not** occur in the first i queries, it is enough to show that

$$\min_{i \leq |\mathcal{A}|} \Pr [\neg E^{i+1} | \neg E^i] \geq 1 - \text{negl}(n) .$$

Indeed, conditioned on $\neg E^i$, the view of $\mathcal{A}$ after the first i queries is information theoretically independent of b . So to show that $\neg E^{i+1}$ occurs with overwhelming probability, it is enough to show that, if it does not occur, $\mathcal{A}$ can reconstruct b from its view in the first i queries. Indeed, if the first $i + 1$ queries contain q, q' satisfying E_2 , then $b = r \oplus r'$, where r is the decryption of q and r' are the first n bits of q' . $\square$

We now claim that, even given the oracles $\mathbb{H}_{\text{sk}}, \mathbb{R}_s, \mathbb{R}_{\text{sk},b,s}$, the encryption scheme using sk is still semantically secure.

Claim 3.3.3 (semantic security). *Let $\mathcal{A}$ be a polysize distinguisher, then:*

$$\left\{ \mathcal{A}^{\mathbb{H}_{\text{sk}}, \mathbb{R}_s, \mathbb{R}_{\text{sk},b,s}}(\text{Enc}_{\text{sk}}(0)) \right\}_{\text{sk},s,b} \approx_c \left\{ \mathcal{A}^{\mathbb{H}_{\text{sk}}, \mathbb{R}_s, \mathbb{R}_{\text{sk},b,s}}(\text{Enc}_{\text{sk}}(1)) \right\}_{\text{sk},s,b} ,$$

where the distributions are also over the randomness of all involved probabilistic functions.

Proof. We first note that by Claims 3.3.1, 3.3.2, we can replace $\mathbb{R}_{\text{sk},b,s}$ (in both distributions) with a random function that is completely independent of sk . Now, the claim follows directly from the CCA-1 security of the encryption scheme, just as in [16, Claim 3.7]. $\square$

Next, let $\mathcal{A}$ be a polysize adversary with oracle access to the (probabilistic) functions

$$\mathbb{A}_{\text{sk},a}, \mathbb{H}_{\text{sk}}, \mathbb{R}_s, \mathbb{R}_{a,b,s}, \mathbb{R}_{\text{sk},b,s}.$$

We define E to be the event that $\mathcal{A}$ produces distinct queries (q, q') to $\mathbb{R}_{a,b,s}$ and $\mathbb{R}_s$, respectively, such that $q = q' \oplus a$.

Claim 3.3.4. $\Pr_{b,\text{sk},s}[E] \leq \text{negl}(n)$.

Proof. First, we note that the event E is testable given a and thus, it is enough to show that the claim holds when PRF_s is replaced by a truly random function R . Now, assume towards contradiction that E occurs with noticeable probability $\varepsilon = \varepsilon(n)$. Then, there exists an $i \leq |\mathcal{A}|$ such that, the event E first occurs in the i 'th query that $\mathcal{A}$ makes with noticeable probability $\varepsilon/|\mathcal{A}|$. We consider an hybrid experiment where, for the first $i-1$ queries that $\mathcal{A}$ makes, $\mathbb{R}_{a,b,R}$ is replaced with an independent random function R' . The view of $\mathcal{A}$ still has the same distribution because the first i answers to $\mathbb{R}_{a,b,R}$ are uncorrelated to the answers of $\mathbb{R}_R$. The latter is true since the queries are not correlated by a , and they're also uncorrelated with the answers of $\mathbb{R}_{\text{sk},b,R}$, which can be thought of as an independent function, because it has different input length.

Thus, in this hybrid experiment, E still occurs for the first time in the i 'th query with probability $\varepsilon/|\mathcal{A}|$. Now, after replacing $\mathbb{R}_{a,b,R}$ with an independent random function R' , for the first $i-1$ queries, we can think about a new adversary $\mathcal{A}_i$ that halts after making the i 'th query. By semantic security Claim 3.3.3 (and a standard hybrid argument), we can replace the oracle $\mathbb{A}_{\text{sk},a}$ with a new oracle $\mathbb{A}_{\text{sk},0^n}$, while affecting the probability that E occurs only by a negligible amount. Now, the view of $\mathcal{A}_i$ is information theoretically independent of a , and thus the probability that it outputs two queries q, q' such that $a = q \oplus q'$ cannot be noticeable, leading to the required contradiction. $\square$

Putting things together Now, for any adversary $\mathcal{A}$, with oracle access to the functions

$$\mathbb{A}_{\text{sk},a}, \mathbb{H}_{\text{sk}}, \mathbb{R}_s, \mathbb{R}_{a,b,s}, \mathbb{R}_{\text{sk},b,s} ,$$

we can first replace all applications of PRF_s with a truly random function R ; then, by Claim 3.3.4, we can replace $\mathbb{R}_{a,b,R}$ with an independent random function R' ; now, we can invoke semantic security (Claim 3.3.3) to replace $\mathbb{A}_{\text{sk},a}$ with $\mathbb{A}_{\text{sk},0^n}$, and finally we can invoke again Claims 3.3.1, 3.3.2, to replace $\mathbb{R}_{\text{sk},b,s}$ with another independent random function R'' . Overall, each change affects the probability that $\mathcal{A}$ outputs b , only by a negligible amount; however, the view of $\mathcal{A}$ in the final hybrid is information theoretically independent of b , and thus, $\mathcal{A}$ cannot output b with noticeable probability.

Deducing unlearnability of f_k Recall that, in the function f_k , $\mathbb{A}_{\text{sk},a}, \mathbb{H}_{\text{sk}}$ are replaced with invoker randomizable deterministic n -fold variants. However, the view of an adversary interacting with f_k can be simulated from the above oracles. Indeed, first note that we can replace $\mathbb{A}_{\text{sk},a}, \mathbb{H}_{\text{sk}}$ by their invoker-randomizable deterministic variants: every new call (y, r) to $\mathbb{G}_{\text{key},s'}$ is answered according to probabilistic oracle $\mathbb{G}_{\text{key}}(y)$, and repeated calls are answered consistently. Then, we can replace any oracle $\mathbb{G}$ with its n -fold variant $\widehat{\mathbb{G}}$: every call to $\widehat{\mathbb{G}}$ is replaced by n calls to $\mathbb{G}$.

This completes the proof of Lemma 27. $\square$

3.3.4 Non-Black-Box Learnability

We now prove that Construction 3.3.1 is non-black-box learnable in the sense of Definition 18. The high-level ideas behind the proof are given in Section 3.1.6.

We describe the extraction procedure. As a first step, we describe an extractor that is only required to work for circuits that perfectly compute each one of the underlying functions, and where $\mathbb{A}_{\text{sk},a,s'}, \mathbb{H}_{\text{sk},s'}$ are given in their 1-fold version. Then, we will explain how to generalize this extractor for the case of faulty circuits, where $\mathbb{A}_{\text{sk},a,s'}^\Delta, \mathbb{H}_{\text{sk},s'}^\Delta$ are in their n -fold version.

Construction 3.3.2. *Let C be a circuit that perfectly implements a function $f_k \in \mathcal{F}$, consisting of circuits $\left\{ \mathbb{A}_{\text{sk},a,s'}^\Delta, \mathbb{H}_{\text{sk},s'}^\Delta, \mathbb{R}_s^\Delta, \mathbb{R}_{a,b,s}^\Delta, \mathbb{R}_{\text{sk},b,s}^\Delta \right\}$ that perfectly implement each of f_k 's underlying functions. Let $\mathbb{A}_{\text{sk},a,s'}^\Delta(U)$ and $\mathbb{H}_{\text{sk},s'}^\Delta(c, c', \odot; U)$ denote the distributions*

of $\mathbb{A}_{\text{sk},a,s'}^\Delta(r)$ and $\mathbb{H}_{\text{sk},a,s'}^\Delta(\mathbf{c}, \mathbf{c}', \odot, r)$, when invoked with uniform invoker randomness r . The extractor $\mathcal{E}$ works according to the following steps:

Step 1: obtain an encryption $\mathbf{c}_a$ of a from $\mathbb{A}_{\text{sk},a,s'}^\Delta(U)$.

Step 2: construct a circuit $C_{a,b}$ that maps a to b , and homomorphically evaluate it on $\mathbf{c}_a$, using $\mathbb{H}_{\text{sk},s'}^\Delta(\cdot; U)$, to obtain an encryption $\mathbf{c}_b$ of b . The circuit $C_{a,b}$ is constructed according to Algorithm (1).

Input: $\tilde{a}$ (allegedly $\tilde{a} = a$)

- 1: sample $r \leftarrow \{0, 1\}^n$
- 2: obtain $\text{ans} \leftarrow \mathbb{R}_s^\Delta(r)$
- 3: obtain $\text{ans}' \leftarrow \mathbb{R}_{a,b,s}^\Delta(\tilde{a} \oplus r)$
- 4: **return** $\text{ans} \oplus \text{ans}'$

Algorithm 1: The a to b circuit - $C_{a,b}$

By the definition of the oracles $\mathbb{R}_{a,b,s}^\Delta(\tilde{a} \oplus r) \oplus \mathbb{R}_s^\Delta(r) = b \oplus \text{PRF}_s(a \oplus \tilde{a} \oplus r) \oplus \text{PRF}_s(r)$, which is indeed b when $\tilde{a} = a$.

Step 3: transform the encryption $\mathbf{c}_b$ of b into b according to the procedure C_b , given by Algorithm (2).

Input: An encryption $\mathbf{c}$ of $\tilde{b}$ (allegedly $\tilde{b} = b$)

- 1: sample $r \leftarrow \{0, 1\}^n$
- 2: using $\mathbb{H}_{\text{sk},s'}^\Delta(\cdot; U)$, homomorphically compute from $\mathbf{c}$ a new encryption $\mathbf{c}'$ of $\tilde{b} \oplus r$
- 3: obtain $\text{ans} \leftarrow \mathbb{R}_{\text{sk},b,s}(\mathbf{c}')$
- 4: obtain $\text{ans}' \leftarrow \mathbb{R}_s(r || \text{pub}(\mathbf{c}'))$
- 5: **return** $\text{ans} \oplus \text{ans}'$

Algorithm 2: The b decoding procedure - C_b

By the definition of the oracles $\mathbb{R}_{\text{sk},b,s}(\mathbf{c}') \oplus \mathbb{R}_s(r || \text{pub}(\mathbf{c}')) = b \oplus \text{PRF}_s(b \oplus \tilde{b} \oplus r || \text{pub}(\mathbf{c}')) \oplus \text{PRF}_s(r || \text{pub}(\mathbf{c}'))$, which is indeed b when $\tilde{b} = b$.

Extraction from circuits with detectable errors We now move to describe our actual extractor $\hat{\mathcal{E}}$ that can extract b , even from a circuit C that approximates the underlying functions related to $f_k \in \mathcal{F}$, and where $\mathbb{A}_{\text{sk},a,s'}^\Delta, \mathbb{H}_{\text{sk},s'}^\Delta$ are given in their n -fold versions. The distributions on which the approximation holds are described following the construction, and are not needed for the description of the extractor, but only for its analysis.

Construction 3.3.3. Let C be a circuit that perfectly implements a function $f_k \in \mathcal{F}$, consisting of circuits $\{\hat{\mathbb{A}}_{\text{sk},a,s'}^\Delta, \hat{\mathbb{H}}_{\text{sk},s'}^\Delta, \mathbb{R}_s^\Delta, \mathbb{R}_{a,b,s}^\Delta, \mathbb{R}_{\text{sk},b,s}^\Delta\}$ that perfectly implement each of f_k 's underlying functions. Let $\hat{\mathbb{A}}_{\text{sk},a,s'}^\Delta(U)$ and $\hat{\mathbb{H}}_{\text{sk},s'}^\Delta(\mathbf{c}, \mathbf{c}', \odot; U)$ denote the distributions

of $\hat{\mathbb{A}}_{\text{sk},a,s'}^\Delta(r_1, \dots, r_n)$ and $\hat{\mathbb{H}}_{\text{sk},a,s'}^\Delta(c, c', \odot, r_1, \dots, r_n)$, when invoked with uniform invoker randomness $r_1, \dots, r_n$.

The extractor $\hat{\mathcal{E}}$ runs a robust “parallel emulation” of $\mathcal{E}$. The emulation follows the same steps as the extraction above only that it augments each one of them. We first describe how homomorphic ciphertext operations are emulated, and then describe how to augment each of the extraction steps.

Homomorphic operations on n -fold ciphertexts In the parallel emulation, $\mathcal{E}$ is given access to the n -fold versions $\hat{\mathbb{A}}_{\text{sk},a,s'}^\Delta$ and $\hat{\mathbb{H}}_{\text{sk},a,s'}^\Delta$ of the cipher producing oracles. These functions output n -fold ciphertexts; namely, a ciphertext for any single bit m is, in fact, an n -fold cipher consisting of n (1-fold) ciphers, each encrypting the bit m . We shall think of n -fold ciphers encrypting n -bit strings as a matrices, where each row is an n -fold encryption of a bit, and each column is a 1-fold encryption of the entire string.

In the parallel emulation, homomorphic operations are done on n -fold bit encryptions. Specifically, given two n -fold ciphers $\vec{c} = (c_1, \dots, c_n)$ and $\vec{c}' = (c'_1, \dots, c'_n)$. For each pair of bit encryptions (c_i, c'_i) , $\hat{\mathcal{E}}$ obtains n samples from $\hat{\mathbb{H}}_{\text{sk},s'}^\Delta(c_i, c'_i, \odot; U)$. Each of the n samples are either $\perp$ or an n -fold cipher. If at least a $\frac{1-\sqrt{\delta}}{2}$ -fraction of the samples are not $\perp$, $\hat{\mathcal{E}}$ continues with the first n -fold cipher as the result of the homomorphic operation; otherwise, $\hat{\mathcal{E}}$ continues to the next pair of ciphers. If none of the pairs produced sufficiently many samples, $\hat{\mathcal{E}}$ aborts.

We now describe emulation Steps 1-3 in the parallel emulation.

Step 1: try at most n times to sample an n -fold encryption M_a of the string a from $\hat{\mathbb{A}}_{a,\text{sk},s'}^\Delta(U)$. If all n samples resulted in $\perp$, $\hat{\mathcal{E}}$ abort.

Step 2: construct a circuit $C_{a,b}$ that maps a to b , and homomorphically evaluate it on M_a to obtain an n -fold encryption M_b of b . The circuit $C_{a,b}$ is constructed according to Algorithm (3), and is an amplified version of the circuit constructed in Algorithm (3).

Step 3: transform the encryption M_b of b into b according to the procedure C_b , given by Algorithm (4), which again is an amplified version of Algorithm (2).

The approximated distributions We define $d = O(1)$ distribution ensembles (or samplers) $\mathcal{D}_1, \dots, \mathcal{D}_d$. For the extractor $\hat{\mathcal{E}}$ to work, the circuit will be required to $(1 - \delta, \mathcal{D}_i)$ -approximate each of these distributions. The distributions are:

Input: $\tilde{a}$ (allegedly $\tilde{a} = a$)

- 1: sample $r_1, \dots, r_n \leftarrow \{0, 1\}^n$
- 2: **for** $j \in [n]$ **do**
- 3: obtain $\text{ans}_j \leftarrow \mathbb{R}_s^\Delta(r_j)$
- 4: obtain $\text{ans}'_j \leftarrow \mathbb{R}_{a,b,s}(\tilde{a} \oplus r_j)$
- 5: **end for**
- 6: **if** $\exists j \in [n]$ such that ans_j and ans'_j are not $\perp$ **then**
- 7: **return** $\text{ans}_j \oplus \text{ans}'_j$
- 8: **else**
- 9: **return** $\perp$
- 10: **end if**

Algorithm 3: The amplified a to b circuit - $C_{a,b}$

Input: An encryption M of $\tilde{b}$ (allegedly $\tilde{b} = b$)
 M has rows $(\vec{c}_1, \dots, \vec{c}_n)$, where $\vec{c}_i$ is an n -fold cipher encrypting $\tilde{b}_i$

- 1: sample $r_1, \dots, r_n \leftarrow \{0, 1\}^n$
- 2: **for** $j \in [n]$ **do**
- 3: homomorphically compute from M a new matrix M' with rows $(\vec{c}'_1, \dots, \vec{c}'_n)$,
 where $\vec{c}'_i$ encrypts the i -th bit of $r'_j = \tilde{b} \oplus r_j$
- 4: **for** $k \in [n]$ **do**
- 5: set $\vec{c}^*_k = (\vec{c}'_1[k], \dots, \vec{c}'_n[k])$ to be the column $M'[k]$,
 which is a 1-fold encryption of r'_j
- 6: obtain $\text{ans}_{j,k} \leftarrow \mathbb{R}_{sk,b,s}(\vec{c}^*_k)$
- 7: obtain $\text{ans}'_{j,k} \leftarrow \mathbb{R}_s(r'_j || \text{pub}(\vec{c}^*_k))$
- 8: **end for**
- 9: **end for**
- 10: **if** $\exists j, k \in [n]$ such that $\text{ans}_{j,k}$ and $\text{ans}'_{j,k}$ are not $\perp$ **then**
- 11: **return** $\text{ans}_{j,k} \oplus \text{ans}'_{j,k}$
- 12: **else**
- 13: **return** $\perp$
- 14: **end if**

Algorithm 4: The amplified b decoding procedure - C_b

- $\mathcal{D}_{\widehat{\mathbb{A}}_{\text{sk},a}}$ consists of n random strings $r_1, \dots, r_n \in \{0, 1\}^{\text{poly}(n)}$ (r_i is thought of as “invoker randomness” for an n -fold encryption of the bit a_i).
- For each operation $\odot$ (from the universal set of gates) and every pair of bits (x, x') , $\mathcal{D}_{\widehat{\mathbb{H}}_{\text{sk}}}^{\odot, x, x'}$ consists of n random strings $r_1, \dots, r_n \in \{0, 1\}^{\text{poly}(n)}$ (for n -fold encryptions of $x \odot x'$), and random 1-fold encryptions (c, c') under sk with underlying plaintexts (x, x') .
- $\mathcal{D}_{\mathbb{R}_s}^n$ consists of a random string $r \in \{0, 1\}^n$.
- $\mathcal{D}_{\mathbb{R}_s}^{n+n^2}$ consists of a random string $r \in \{0, 1\}^n$, and n independent samples from $\text{PubSamp}(1^n)$ of random “public parts of a cipher” as defined in 25.
- $\mathcal{D}_{\mathbb{R}_{a,b,s}}$ consists of a random string $r \in \{0, 1\}^n$.
- $\mathcal{D}_{\mathbb{R}_{\text{sk},b,s}}$ consists of a random 1-fold encryption $(c_1, \dots, c_n)$ to the bits of a random string $r \in \{0, 1\}^n$.

In addition, each of the distributions $\mathcal{D}$ above also includes an index $i_{\mathbb{G}} \in [5]$, indicating which function $\mathbb{G} \in \{\widehat{\mathbb{A}}_{\text{sk},a,s'}, \widehat{\mathbb{H}}_{\text{sk},s'}, \mathbb{R}_s, \mathbb{R}_{a,b,s}, \mathbb{R}_{\text{sk},b,s}\}$ to invoke.

Key-independent sampling Recall that in Definition 18 for d -distribution robustness, we do not require that each one of the distributions $\mathcal{D}_1, \dots, \mathcal{D}_d$ is efficiently samplable independently of the key k . While almost all the above distributions can be sampled independently of k , the distributions of the form $\mathcal{D}_{\widehat{\mathbb{H}}_{\text{sk}}}^{\odot, x, x'}$ may not be samplable without the secret encryption key sk , even given (x, x') ; indeed, we only assume sampling of ciphers for random plaintexts (Definition 26). We do require, however, that the distributions $\{\mathcal{D}_i\}$ are jointly key-independent (Definition 19), implying that, for a random $i^* \leftarrow [d]$, we can sample from $\mathcal{D}_{i^*}$, independently of k (in Definition 19 this sampler is denoted by $\mathcal{D}^*$). In our case, this is satisfied due to random ciphertext generation: instead of sampling at random $x, x' \in \{0, 1\}$, and then sampling encryptions (c, c') of (x, x') , we can directly sample two independent ciphers of random bits using RanSamp .

Remark 28 (One uniform distribution). *In the general Definition 10 of unobfuscatable functions, we allow the distribution ensemble $\mathcal{D}$ to be an arbitrary samplable distribution. As mentioned in Remark 12, we can consider a more strict (but natural) definition where $\mathcal{D}$ is required to be the uniform distribution. Our constructions can*

actually achieve this notion. Indeed, the input distribution in our eventual construction is the distribution $\mathcal{D}^*$ as defined in Definition 19, with respect to $\mathcal{D}_1, \dots, \mathcal{D}_d$ above. The distribution $\mathcal{D}^*$ can be made uniform.

Specifically, the distributions $\mathcal{D}_1, \dots, \mathcal{D}_d$, using the symmetric key encryption described in Section 3.3.1, the sampler for random public parts **PubSamp** and the sampler for random ciphers **RanSamp** both output uniformly random strings. So each distribution $\mathcal{D}_j$ described above simply consists of a random strings (of some length) and an index $i_{\mathbb{G}} \in [5]$. In particular, by appropriately padding with extra randomness, we can think of each of the n blocks of $\mathcal{D}^*$ as a uniform string representing a random $i_{\mathbb{G}} \in [5]$, plus an extra random string representing the input to $\mathbb{G}$. (There is an extra technical issue that can be taken care of: the index $i_{\mathbb{H}}$ is output by several distributions $\widehat{\mathcal{D}}_j$, and not just one, which may skew the uniformity of $i_{\mathbb{G}}$; however, this can be taken care of by artificially adding more indices to represent $i_{\mathbb{H}}$.)

We now move on to proving that the construction described in this section is non-black-box learnable.

Lemma 29. $\mathcal{F}$ given by Construction 3.3.1 is non-black-box learnable in the sense of Definition 18.

Proof. We start by giving a roadmap to the proof; the high-level ideas behind the proofs are described in the introduction.

Roadmap for the proof We analyze the success of the extractor given a circuit that successfully answers (does not output $\perp$ or errs) with probability $1 - \delta$, for each of the distributions $\mathcal{D}_1, \dots, \mathcal{D}_d$ above, where $\delta < 2^{-5}$ is a constant. Our first step is to show the completeness of homomorphic evaluation; namely, that when the extractor performs homomorphic operations (in its parallel emulation mode), it will almost never get stuck, and will almost always obtain a new n -fold cipher representing the result of the homomorphic computation. At the second step, we will prove that the circuit $C_{a,b}$ constructed by the extractor indeed performs properly; that is, it maps a to b . Finally, we will show that the procedure C_b , when given the n -fold encryption of b produced by $\widehat{\mathcal{E}}$, successfully outputs b .

Throughout the analysis, we will condition on the event that the circuit C (that we extract from) does not make any undetectable errors (i.e., $C(q) \notin \{f_k(q), \perp\}$) on any

query q sampled by the extractor for one of the functions implemented by the circuit. Recall, that the probability of an undetectable error is $\nu(n)$, and the total number of queries made by the extractor is bounded by some $\text{poly}(|C|, n)$. Thus, this condition is violated and may cause extraction failure with probability at most $\nu(n) \cdot (|C| \cdot n)^{O(1)}$, which is counted within the failure probability of the extractor.

Completeness of the homomorphic evaluation phase We call a pair of 1-fold ciphers (c, c') encrypting a pair of bits x and x' **good** for $\odot$ if

$$\Pr \left[\widehat{\mathbb{H}}_{\text{sk}}^\Delta(c, c', \odot; U) \neq \perp \right] \geq 1 - \sqrt{\delta} ,$$

where $\widehat{\mathbb{H}}_{\text{sk}}^\Delta(c, c', \odot; U)$ is as defined in Construction 3.3.3. We say that a pair of n -fold ciphers $(\vec{c}, \vec{c}')$, each encrypting a pair of bits x and x' is **good** for $\odot$ if there exists $i \in [n]$ such that the 1-fold pair $(\vec{c}[i], \vec{c}'[i])$ is **good** for $\odot$. We say that a pair of n -fold ciphers is **good** if it is **good** for any operation $\odot$.

Since the circuit $\widehat{\mathbb{H}}_{\text{sk}}^\Delta$ answers $\mathcal{D}_{\widehat{\mathbb{H}}_{\text{sk}}^\Delta}^{\odot, x, x'}$, where (c, c') are random, with probability $1 - \delta$, it holds that at least a $(1 - \sqrt{\delta})$ -fraction of the cipher-pairs encrypting (x, x') are **good** for $\odot$. Therefore, a random pair of n -fold ciphers is **not good** with probability at most $\delta^{\frac{n}{2}} \cdot d$ (indeed, the number of distributions d is a bound on the number of operations $\odot$). We will use this to show:

Claim 3.3.5. *The extractor $\widehat{\mathcal{E}}$ does not abort during the homomorphic evaluation phase, except with probability $2^{-\Omega(n)} \cdot (|C| \cdot n)^{O(1)}$.*

Proof. We first claim that, except with probability $2^{-\Omega(n)} \cdot (|C| \cdot n)^{O(1)}$, for every two n -fold ciphers $\vec{c}$ and $\vec{c}'$ that are produced during the execution of $\widehat{\mathcal{E}}$ (at different times), the pair $(\vec{c}, \vec{c}')$ is **good**. Indeed, note that any two executions of the circuit $\widehat{\mathbb{H}}_{\text{sk}}^\Delta$ produce a pair $(\vec{c}, \vec{c}')$ that is not **good** only if the corresponding two executions of the function $\widehat{\mathbb{H}}_{\text{sk}}^\Delta$ (that, unlike its approximation, never outputs $\perp$), given the same inputs, output such a pair. Moreover, since $\widehat{\mathbb{H}}_{\text{sk}}^\Delta$ is invoker randomizable, the output of any two executions is uniformly distributed over all pairs of n -fold ciphers, and, therefore, is not **good** only with negligible probability $\delta^{\frac{n}{2}} \cdot d$; in particular, this is also the case for any pair produced by an execution of the circuit $\widehat{\mathbb{H}}_{\text{sk}}^\Delta$. The same argument holds for any pair of n -fold ciphers generated by the circuit $\widehat{\mathbb{A}}_{a, \text{sk}}^\Delta$ (or for mixed pairs created by the two circuits). The bound concrete $2^{-\Omega(n)} \cdot (|C| \cdot n)^{O(1)}$ bound is obtained by taking a union

bound over at most $(|C| \cdot n)^{O(1)}$ calls made to the above oracles.

We now claim that, except with probability $2^{-\Omega(n)} \cdot (|C| \cdot n)^{O(1)}$, the extractor does not abort during the homomorphic evaluation phase. We condition on the (overwhelmingly often event) that the extractor only runs homomorphic operations for good pairs. First, note that, during Step 1, $\widehat{\mathcal{E}}$ aborts only if all n samples from $\widehat{\mathbb{A}}_{a, \text{sk}, s'}^\Delta(U)$ are $\perp$. Thus, because the circuit $\widehat{\mathbb{A}}_{\text{sk}, a}^\Delta$ answers on $\widehat{\mathcal{D}}_{\mathbb{A}_{\text{sk}, a}}$ with probability $1 - \delta$, $\widehat{\mathcal{E}}$ aborts with probability at most δ^n . In any other step, when $\widehat{\mathcal{E}}$ is running a homomorphic operation $\odot$ for a good pair of n -fold ciphers $(\vec{c}, \vec{c}')$, there always exists some $i \in [n]$ such that the pair of ciphers $(\vec{c}[i], \vec{c}'[i])$ is good for $\odot$. Therefore, each of the n samples drawn from $\widehat{\mathbb{H}}_{\text{sk}, s'}^\Delta(\vec{c}[i], \vec{c}'[i], \odot; U)$ are not $\perp$ with probability at least $1 - \sqrt{\delta}$. Recall, that $\widehat{\mathcal{E}}$ aborts only if it fails to obtain more than $\frac{1-\sqrt{\delta}}{2}$ -fraction of the samples in all iterations; however, this would occur in the good iteration i only with an exponentially small probability $2^{-\Omega(n)}$. Again we take a union bound over at most $(|C| \cdot n)^{O(1)}$ oracle calls. $\square$

Completeness of the circuit $C_{a,b}$ We now show that the probabilistic circuit $C_{a,b}$ constructed by the extractor (almost always) returns b on input a as required.

Claim 3.3.6. $\Pr_{C_{a,b}}[C_{a,b}(a) \neq b] \leq 2^{-\Omega(n)}$.

Proof. Recall that $C_{a,b}$, given a as input, samples n random strings $r_1, \dots, r_n$, and queries $\mathbb{R}_s(r_i)$, $\mathbb{R}_{a,b,s}(a \oplus r_i)$ to obtain b ; this succeeds if both oracles did not output $\perp$ which is the case with probability $1 - 2\delta$. The probability that none of the n trials succeeds is at most $(1 - 2\delta)^n$. $\square$

Completeness of the procedure C_b We now show that the procedure C_b (almost always) returns b on the n -fold encryption of b produced by the extractor.

Claim 3.3.7. *Let $\vec{c}$ be the n -fold encryption of b produced by $\widehat{\mathcal{E}}$, then*

$$\Pr_{C_b}[C_b(\vec{c}) = b] \geq 1 - 2^{\Omega(n)} \cdot (|C| \cdot n)^{O(1)}.$$

Proof. Throughout, we condition on the (overwhelmingly often) event that homomorphic operations do not fail, which is the cause for the $(|C| \cdot n)^{O(1)}$ factor in the final probability. Let $\mathbf{c}$ be a random 1-fold encryption of a random n -bit string r , then $\mathbf{c}$ is answered by $\mathbb{R}_{\text{sk}, b, s}$ with probability at least $1 - \delta$. For such a random $\mathbf{c}$, let

$r(c) = \text{Dec}_{\text{sk}}(c)$ be the underlying plaintext, and let $\text{pub}(c)$ be the public part of c . Then $(b \oplus r(c), \text{pub}(c))$ is answered by $\mathbb{R}_s$ with probability at least $1 - \delta$; indeed, a random cipher of a random string, such as c , induces a random pair $(b \oplus r(c), \text{pub}(c))$. It follows that with probability at least $1 - 2\delta$ over the choice of c both of the above queries are answered, we call such c - **good**.

We say that r is **good** if a random 1-fold encryption c of r is **good** with probability $1 - \sqrt{2\delta}$; in particular, we know that there is a $(1 - \sqrt{2\delta})$ -fraction of **good** r 's.

Next, for a fixed r , let the matrix M be an n -fold encryption of r , with rows $(\vec{c}_1, \dots, \vec{c}_n)$, such that each $\vec{c}_i$ is an n -fold cipher encrypting the bit r_i . We say that such a matrix M is **good** if one of its columns $M[j] = (\vec{c}_1[j], \dots, \vec{c}_n[j])$ is a **good** (1-fold) encryption of the string r (as defined above). By the previous paragraph, it follows that if r is **good**, then a random matrix M encrypting it is **not good** with probability at most $(2\delta)^{\frac{n}{2}}$.

Now, recall that, for random $(r_1, \dots, r_n)$, C_b generates for each $r = r_j$ a matrix M' encrypting $r' = b \oplus r$, where the i -th row of M' is an n -fold cipher encrypting the bit r'_i . Fix any such row of the matrix M' , and recall that this row is generated homomorphically from a pair of ciphers (c, c') . We claim that, except with negligible probability, $\Pr \left[\widehat{\mathbb{H}}_{\text{sk}, s'}^\Delta(c, c', \odot; U) \neq \perp \right] \geq \frac{1-\sqrt{\delta}}{4}$; otherwise, the homomorphic evaluation procedure would skip (c, c') with overwhelming probability. Indeed, the homomorphic procedure always evaluates the above probability, and skips any pair of ciphers whose estimate is worst than $\frac{1-\sqrt{\delta}}{2}$. Next, note that the output of the (non-aborting) invoker-randomizable function $\widehat{\mathbb{H}}_{\text{sk}, s'}^\Delta(c, c', \odot; U)$ is uniformly distributed over all n -fold ciphers of r'_i . It follows that the i -th row of M' sampled from the circuit $\widehat{\mathbb{H}}_{\text{sk}, s'}^\Delta(c, c', \odot; U)$ is uniformly distributed over at least a $\frac{1-\sqrt{\delta}}{4}$ -fraction of all n -fold ciphers encrypting the bit r'_i . Since this holds for every i , the matrix M' is uniformly distributed over a $\left(\frac{1-\sqrt{\delta}}{4}\right)^n$ -fraction of all random cipher matrices encrypting r' .

It now follows, that for any **good** r' , the matrix M' encrypting $r' = r \oplus b$ produced by C_b is **good**, except with probability $\frac{(2\delta)^{\frac{n}{2}}}{\left(\frac{1-\sqrt{\delta}}{4}\right)^n} \leq \left(\frac{\sqrt{32\delta}}{1-\sqrt{\delta}}\right)^n \leq 2^{-\Omega(n)}$, for our choice of $\delta < 2^{-5}$. Since a $(1 - \sqrt{2\delta})$ -fraction of the r 's are **good**, and our extractor tries n such r 's, with overwhelming probability the extractor will go through a **good** matrix M' with probability $1 - 2^{-\Omega(n)}$. In such a case, the answers $\text{ans}_{j,k}$ and $\text{ans}'_{j,k}$ it obtains from the oracles $\mathbb{R}_{\text{sk}, b, s}$ and $\mathbb{R}_s$ are not $\perp$, and $b = \text{ans}_{j,k} \oplus \text{ans}'_{j,k}$ as required. $\square$

This concludes the proof of Lemma 29. □

3.3.4.1 Necessity of One-Way Functions

We show that one-way functions are not only sufficient for constructing robust unobfuscatable functions, but also necessary given the natural requirement of a unique-witness unlearnable relation noted in Remark 14.

Lemma 30. *Robust unobfuscatable functions with unique witness unlearnable relations imply one-way functions.*

Proof. Let $\mathcal{F} = \{f_k : \{0, 1\}^n \rightarrow \{0, 1\}^*\}_{k \in \{0, 1\}^n, n \in \mathbb{N}}$ be a family of robust unobfuscatable functions. For simplicity, let us also assume that the family is $\frac{2}{3}$ -robust with respect to the uniform distribution ensemble $\mathcal{D} = \mathcal{U}$. (By Claim 17, $\frac{2}{3}$ -robustness can be amplified to full robustness, so this is WLOG. Also, the proof easily generalizes to an arbitrary samplable distribution ensemble $\mathcal{D}$). We define a one-way function $\varphi = \{\varphi_n\}$ as follows: $\varphi_n : \{0, 1\}^{n(m+1)} \rightarrow \{0, 1\}^*$, and

$$(k, x_1, \dots, x_m) \xrightarrow{\varphi_n} (f_k(x_1), \dots, f_k(x_m), x_1, \dots, x_m) ,$$

where $m(n) = 2n$.

We next show that φ is one-way. For any $k \in \{0, 1\}^n$, we denote by z_k its corresponding unique unlearnable secret. First note that for any (k, k') such that $z_k \neq z_{k'}$, the agreement between f_k and $f_{k'}$ is at most $2/3$. Indeed, if that was not the case, we could construct a circuit $C_{k \cap k'}$ that given input x returns $f_k(x)$, only if $f_k(x) = f_{k'}(x)$, and otherwise returns $\perp$. This circuit computes both functions with probability at least $2/3$ (without making errors); however, in this case the robust extraction guarantee, implies that the extractor outputs $z \in \mathcal{R}_{\mathcal{F}}(k) \cap \mathcal{R}_{\mathcal{F}}(k') = \{z_k\} \cap \{z_{k'}\}$, which means that $z_k = z_{k'}$.

Thus we can deduce that for any $k \in \{0, 1\}^n$:

$$\Pr_{x_1, \dots, x_m} \left[\exists k' \left| \begin{array}{l} z_k \neq z_{k'} \\ \forall i \in [m] : f_k(x_i) = f_{k'}(x_i) \end{array} \right. \right] \leq |\{k' \in \{0, 1\}^n\}| \cdot \left(\frac{2}{3}\right)^m \leq \left(\frac{8}{9}\right)^n .$$

It follows that any algorithm that inverts φ with noticeable probability, outputs

k' such that $z_{k'} = z_k$; in particular, such an algorithm directly implies a learner that breaks the black-box unlearnability of $\mathcal{F}$ with respect to its unlearnable relation $\mathcal{R}_{\mathcal{F}}$. This learner would simply query its oracle f_k on m random points, run the inverter to obtain k' and compute $z_{k'} = z_k$. $\square$

Remark 31. *We note that the above lemma also holds if we consider a weaker form of robust unobfuscatable Turing machine families, whereas non-robust unobfuscatable Turing machine families can be constructed without any computational assumptions (see [16]). In addition, the above also holds given a weaker inefficient extraction guarantee.*

3.3.5 More Efficient Extraction from Fully Homomorphic Encryption

In this section, we discuss the running time of the extractor and its effect on applications. We analyze the running of the extractor corresponding to Construction 3.3.1, and also present an augmented construction with more efficient extraction based on fully homomorphic encryption.

The extractor $\widehat{\mathcal{E}}$ given by Construction 3.3.3 evaluates each of the functions $\mathbb{R}_{\mathbf{s}}^{\Delta}, \mathbb{R}_{a,b,\mathbf{s}}^{\Delta}, \mathbb{R}_{\mathbf{sk},b,\mathbf{s}}^{\Delta}$, implemented by the input circuit C , $\text{poly}(n)$, for some fixed polynomial poly . However, some of these evaluations are “homomorphic evaluations” of the circuit C on certain encryptions. Since each homomorphic operation is performed by evaluating the function $\mathbb{H}_{\mathbf{sk},\mathbf{s}'}^{\Delta}$ implemented by C , we get that the total running of $\widehat{\mathcal{E}}$ is $\text{poly}(n) \cdot |C|^2$. (In particular, after applying the transformation from Lemma 17 to obtain robust unobfuscatable functions for an arbitrary approximation parameter ε , the running time of the extractor is $\text{poly}(n) \cdot (|C|/\varepsilon)^2$.)

As discussed in Section 3.1.5, and will be further discussed in Section 3.5.4.2, improving the ratio between the running time of $\widehat{\mathcal{E}}$ and $|C|$ has applications to reducing the round complexity of resettably-sound and simultaneously-resettable ZK. By replacing the symmetric encryption in Construction 3.3.1 with fully homomorphic encryption we can improve the running time of $\widehat{\mathcal{E}}$ to $\text{poly}(n) \cdot |C|$; namely, we can make the dependency on $|C|$ linear instead of quadratic.

The main idea is that now $\widehat{\mathcal{E}}$ is able to perform each homomorphic operation on its own in time $\text{poly}(n)$, without evaluating the entire circuit C , implementing the function $\widehat{\mathbb{H}}_{\mathbf{sk},\mathbf{s}'}^{\Delta}$.

The augmented construction We start by overiewing the basic properties needed by the encryption scheme, compared to those of the previous one. First, we require that the scheme is fully-homomorphic and *rerandomizable* in the following sense:

Definition 32 (Rerandomizable encryption). *A bit encryption scheme $(\text{Gen}, \text{Enc}, \text{Dec})$ is rerandomizable if there exist an efficient ciphertext rerandomization algorithm ReRan such that and a public ciphertext sampling algorithm RanSamp , such that for every sequence of ciphers (encrypting a bit) and secret keys $\{c_n, \text{sk}_n\}_{n \in \mathbb{N}}$:*

$$\{\text{ReRan}(c_n)\}_{n \in \mathbb{N}} \approx_s \{\text{RanSamp}(\text{Dec}_{\text{sk}_n}(c_n))\}_{n \in \mathbb{N}} .$$

(Statistical indistinguishability can be naturally relaxed to computational.)

Known fully-homomorphic encryptions schemes from the literature (e.g., [46]) satisfy such a rerandomization property. This is usually done to guarantee function privacy, and is achieved by adding to evaluated ciphertexts a sufficient amount of noise. In our case, we would like the noise to be taken from a fixed distribution that does not depend on the evaluated circuit (in our context, the circuit we extract from); this can be ensured using the standard noise-control techniques, such as bootstrapping [88].

In our modified construction, the learner will not have access to a function that performs homomorphic ciphertext operations and therefore we no longer require that the encryption is CCA-1. Additionally, our previous requirement of random ciphertext generation is subsumed by the above rerandomization property (by publishing encryptions of 0 and 1 that can be rerandomized).

Recall that, in Construction 3.3.1, we required that the underlying encryption scheme is decomposable (according to Definition 25). We will make the same requirement for the augmented construction. We next show that, in fact, any homomorphic encryption scheme can be also made decomposable, while preserving homomorphism and rerandomization.

Construction 3.3.4 (Making homomorphic encryption decomposable). *Let $(\text{Gen}, \text{Enc}, \text{Dec}, \text{Eval})$ be a fully homomorphic encryption scheme. We construct a decomposable fully homomorphic encryption scheme $(\text{Gen}', \text{Enc}', \text{Dec}', \text{Eval}')$ as follows:*

- Gen' is identical to Gen .

- Enc' on a secret key sk and plaintext $b \in \{0, 1\}$, samples a random bit r from $\{0, 1\}$ and outputs $\text{Enc}_{\text{sk}}(r) || b \oplus r$.
- Dec' on a secret key sk and ciphertext $c || r$, outputs $r \oplus \text{Dec}_{\text{sk}}(c)$.
- Eval' on a pair of ciphertexts $c_1 || r_1$ and $c_2 || r_2$ encrypting b_1 and b_2 , and an operation $\odot$, samples a bit $r \leftarrow \{0, 1\}$ and outputs $c || r$ where the ciphertext c is computed homomorphically (using Eval) from (c_1, c_2, r_1, r_2, r) to encrypt $b_1 \odot b_2 \oplus r$.
- If the original scheme has a rerandomization algorithm ReRan , the new scheme has a rerandomization algorithm ReRan' that on input ciphertext $c_1 || r_1$, samples a random bit r from $\{0, 1\}$, homomorphically computes a cipher c such that $\text{Dec}(c_1 || r_1) = \text{Dec}(c || r)$ and outputs $\text{ReRan}(c) || r$.

The algorithm pub on input ciphertext $c || r$ outputs c . (As remarked in Section 3.3.1, the algorithm PubSamp can be implemented by sampling a random cipher of a random plaintext and applying pub .)

Theorem 33. Assuming rerandomizable homomorphic encryption, there exist a family of robust unobfuscatable functions, where the running time of the extractor $\mathcal{E}(C, 1^n)$ is $\text{poly}(n) \cdot |C|$.

Construction 3.3.5 (Robust unobfuscatable functions with efficient extraction). The construction is a variant of Construction 3.3.1 with the following modifications:

- The encryption scheme is rerandomizable and fully homomorphic.
- The function f_k will no longer evaluate the function $\hat{\mathbb{H}}_{\text{sk}}$.
- When making the function $\mathbb{A}_{\text{sk},a}$ deterministic, a standard PRF can be used (instead of one that is invoker randomizable). In addition, the n -fold $\hat{\mathbb{A}}_{\text{sk},a}$ can be replaced with the 1-fold version $\mathbb{A}_{\text{sk},a}$.

Proof sketch of Theorem 33. The proof of black-box unlearnability is similar to the proof of Lemma 27. In the proof of non-black-box learnability, the main modification we introduce to the extractor given by Construction 3.3.3 is in the emulation of ciphertext operations. Specifically, ciphertexts are no longer represented by n -fold ciphers; instead, values of intermediate wires in the homomorphic evaluation performed by $\hat{\mathcal{E}}$ are represented by a single ciphertext, and homomorphic operations are performed using

Eval (instead of using the function $\widehat{\mathbb{H}}_{sk,s'}^\Delta$). Also, the circuit C_b is given one encryption of b instead of n encryptions. C_b will now use the ciphertext rerandomization algorithm ReRan to generate n random encryptions of b . $\square$

3.4 Publicly-Verifiable Robust Unobfuscatable Functions

In this section, we define publicly-verifiable robust unobfuscatable functions. In Section 3.4.1, we show how to construct them from robust unobfuscatable functions (using a general compiler). We also show that such functions imply error-robust unobfuscatable functions. Throughout, we shall simply call them verifiable (rather than publicly-verifiable).

At high-level, a verifiable robust unobfuscatable family $\mathcal{F}$ is associated with a key generation algorithm $\text{Gen}_{\mathcal{F}}$ that samples a secret key k and a (public) verification key vk . The verification key has two purposes:

- **Public verification of the unlearnable relation.** vk allows verifying any candidate witness for the unlearnable relation $\mathcal{R}_{\mathcal{F}}$; namely, given z , it can be efficiently checked whether $\mathcal{R}_{\mathcal{F}}(vk, z) = 1$, whereas in Definition 10, the secret key k is required for verification.
- **Public verification of an image property.** Intuitively, vk is meant to allow verifying that a given value a is indeed the correct evaluation of f_k on some known input q . We generalize the latter so that vk allows to verify a given property of a candidate image a for $f_k(q)$, for a given input q . Specifically, the family is associated with an efficient relation $\text{Ver}_{\mathcal{F}}$, where $\text{Ver}_{\mathcal{F}}(vk, q, a) = 1$ only if the pair (q, a) satisfies a given property with respect to f_k . For example, f_k may consist of two functions (f_{k_1}, f_{k_2}) , and $\text{Ver}_{\mathcal{F}}$ may verify that a pair $(q_1, a_1), (q_2, a_2)$ is such that either $a_1 = f_{k_1}(q_1)$ or $a_2 = f_{k_2}(q_2)$. We will require that $\text{Ver}_{\mathcal{F}}$ has a completeness property, implying that it is always the case that, if $a = f_k(q)$, then $\text{Ver}_{\mathcal{F}}(vk, q, a) = 1$.

The black-box unlearnability property is defined just as for robust unobfuscatable functions, except that it should hold also against learners that are given the verification key vk . The non-black-box learnability property is strengthened: in the definition of robustness, the extractor was required to work for any C that agrees with the function

on a given distribution $\mathcal{D} = \mathcal{D}(1^n)$; now, we require that extraction works, even if the circuit C may never produce $a = f_k(q)$, but does produce a 's such that $\text{Ver}_{\mathcal{F}}(vk, q, a) = 1$ with high-probability over inputs drawn from $\mathcal{D}$. In particular, the circuit C is allowed to output undetectable errors $a \notin \{f_k(q), \perp\}$.

Definition 34 (A verifiable robust unobfuscatable function). *A family of functions $\mathcal{F}$, with efficient key generation algorithm $\text{Gen}_{\mathcal{F}}$, and image verification relation $\text{Ver}_{\mathcal{F}}$, is a verifiable robust unobfuscatable family with respect to $\mathbf{NP}$ relation $\mathcal{R}_{\mathcal{F}}$ and input sampler $\mathcal{D}$ if it is:*

1. **Black-box unlearnable:** *For any poly-size learner L and any $n \in \mathbb{N}$:*

$$\Pr_{(k, vk) \leftarrow \text{Gen}_{\mathcal{F}}(1^n)} [(vk, z) \in \mathcal{R}_{\mathcal{F}} : z \leftarrow L^{f_k}(vk)] \leq \text{negl}(n) .$$

2. **Non-Black-box learnable:** *There exists an efficient extractor $\mathcal{E}$ such that, for any function $\varepsilon = \varepsilon(n) \geq 2^{-\sqrt{n}}$, all large enough $n \in \mathbb{N}$, any secret key and verification key $(k, vk) \in \text{supp}(\text{Gen}_{\mathcal{F}}(1^n))$, and every circuit C such that*

$$\Pr_{q \leftarrow \mathcal{D}(1^n)} [\text{Ver}_{\mathcal{F}}(vk, q, a) = 1 : a = C(q)] \geq \varepsilon ,$$

$\mathcal{E}$ extracts $z \in \mathcal{R}_{\mathcal{F}}(vk)$ from C :

$$\Pr_{\mathcal{E}} [(vk, z) \in \mathcal{R}_{\mathcal{F}} : z \leftarrow \mathcal{E}(C, vk, 1^n, 1^{1/\varepsilon})] \geq 1 - 2^{-\Omega(n)} \cdot |C|^{O(1)} .$$

3. **Verification completeness:** *For any $(k, vk) \in \text{supp}(\text{Gen}_{\mathcal{F}}(1^n))$, and for any q in the domain of f_k , $\text{Ver}_{\mathcal{F}}(vk, q, f_k(q)) = 1$.*

Remark 35 (Extraction against computationally bounded circuits). *Extraction in the above definition is information theoretic, and does not require an explicit bound on the size of the circuit C from which extraction is made. The definition can be naturally relaxed to only support extraction from poly-size circuit families, in which case there will be some $\text{negl}(n)$ error in the extraction probability that depends on the concrete family at hand. (See also Remark 15.)*

We now observe that verifiable robust unobfuscatable functions imply error-robust unobfuscatable functions.

Lemma 36. *Any verifiable robust unobfuscatable function family $\mathcal{G}$ can be transformed into an error-robust unobfuscatable function family $\mathcal{F}$ (according to Definition 11).*

Proof. The transformation from $\mathcal{G}$ to $\mathcal{F}$: a key k' for a function $f_{k'} \in \mathcal{F}$ is a pair (k, vk) consisting of a key k and a corresponding verification key vk for a function g_k . The input sampler $\mathcal{D}$ is the same as for $\mathcal{F}$, and the function $f_{k'}$ is defined as follows: given an input $q \leftarrow \mathcal{D}(1^n)$, $f_{k'}(q)$ returns $(g_k(q), vk)$. The unlearnable relation $\mathcal{R}_{\mathcal{F}}$ is $\{((k, vk), z) : z \in \mathcal{R}_{\mathcal{G}}(vk)\}$.

First, black-box unlearnability of $\mathcal{F}$ follows readily from that of $\mathcal{G}$. We show non-black-box learnability. Let C be any circuit that $(\frac{1}{2} + \varepsilon, \mathcal{D})$ -approximates $f_{k'}$ **even with errors** (where $\mathcal{D} = \mathcal{D}(1^n)$). The extractor $\mathcal{E}_{\mathcal{F}}$ will first obtain $O(n/\varepsilon)$ samples $(q, C(q))$, for $q \leftarrow \mathcal{D}$. It will then identify the single verification key vk such that $k' = (vk, k)$; by a Chernoff bound, this can be done with probability $1 - 2^{-\Omega(n)}$, by taking the majority. $\mathcal{E}_{\mathcal{F}}$ will then construct from C a new circuit C' that $(\frac{1}{2} + \varepsilon, \mathcal{D})$ -approximates f_k in the sense of Definition 34. This is simply done by verifying each and every answer by $\text{Ver}_{\mathcal{F}}(vk, \cdot, \cdot)$. The completeness of $\text{Ver}_{\mathcal{F}}$ says that $\text{Ver}_{\mathcal{F}}$ would only filter out outputs on which C does not agree with the function g_k , implying that

$$\Pr_{q \leftarrow \mathcal{D}(1^n)} [\text{Ver}_{\mathcal{F}}(vk, q, a) = 1 : a = C'(q)] \geq \varepsilon .$$

Thus, the extractor $\mathcal{E}_{\mathcal{F}}$ can run the underlying extractor $\mathcal{E}_{\mathcal{G}}$ and obtain $z \in \mathcal{R}_{\mathcal{G}}(vk)$ with probability $1 - 2^{-\Omega(n)} \cdot |C|^{O(1)}$. $\square$

3.4.1 Constructing Verifiable Robust Families

We now show how to construct verifiable robust families from robust families with a hardcore secret (Definition 21). (Recall that the latter can be constructed, in turn, from any robust family with a unique-witness unlearnable relation, such as the family given by Section 3.3).

Theorem 37. *Assuming trapdoor permutations, any robust unobfuscatable $\mathcal{G}$ with respect to a hardcore family $\{\mathcal{HC}_n\}$, can be compiled into a verifiable robust family $\mathcal{F}$ (according to Definition 34).*

Proof. We prove the theorem in two steps: we first show how to augment the key generation procedure of $\mathcal{G}$ so that it would also produce a verification key and define

a corresponding (publicly-verifiable) unlearnable relation. $\mathcal{G}$ will be robust also with respect to the new relation, but still only against detectable errors. (This does not involve changing the functions in $\mathcal{G}$ themselves, but only the key generator.)

Then, in the second step, we show how to transform $\mathcal{G}$ (with its augmented generator) into a new family that also has a public-verifiable image property, and will now be robust in the stronger sense of Definition 34 (i.e., extraction is guaranteed whenever the image property is satisfied with high probability).

Step 1 - a family with a publicly-verifiable unlearnable relation. Given a family $\mathcal{G}$ with a hardcore family $\{\mathcal{HC}_n\}$, we define a new key generator $\text{Gen}_{\mathcal{F}}(1^n)$ that samples a (private) key $k \leftarrow \{0,1\}^n$ (defining a function $g_k \in \mathcal{G}$), together with a public key $vk = (h, y)$, where $h \leftarrow \mathcal{HC}_n$ is a randomly chosen hardcore function, and $y = \varphi(h(k))$ for a one-way function φ . We then define the publicly verifiable unlearnable relation to be

$$\mathcal{R}_{\mathcal{G}} = \{(vk, z) \mid vk = (h, y), y = \varphi(z)\} \quad .$$

Claim 3.4.1. *$\mathcal{G}$ is black-box unlearnable with respect to the relation $\mathcal{R}_{\mathcal{G}}$ in the sense of Definition 34; namely, even given a verification key. Also, $\mathcal{G}$ is non-black-box learnable in the sense of Definition 10 with respect to the relation $\mathcal{R}_{\mathcal{G}}$.²*

Proof. First, we show that a randomly chosen $g_k \in \mathcal{G}$ is black-box unlearnable with respect to the relation $\mathcal{R}_{\mathcal{G}}$, even given a corresponding verification key vk . Recall that $h(k)$ is pseudo random, even given black-box access to g_k and h . Thus, any learner that manages to satisfy $\mathcal{R}_{\mathcal{G}}$ would also manage to do so had we given it $\varphi(u)$, for an independent random u , which contradicts the one-wayness of φ .

Non-black-box learnability of $\mathcal{G}$ with respect to the relation $\mathcal{R}_{\mathcal{G}}$ follows directly from the unlearnability of $\mathcal{G}$ with respect to the hardcore family $\mathcal{HC}$ (as given by Definition 21. Indeed, given a circuit that $(\varepsilon, \nu, \mathcal{D})$ -approximates g_k , and $vk = (h, y)$, we can invoke the extractor given by Definition 21 to obtain $h(k)$, and thus obtain a pre-image in $\varphi^{-1}(y)$ as required. $\square$

²Formally, non-black-box learnability in Definition 10 addresses the ability to learn a witness for a given privately-verifiable relation. Here we think about the natural analog: we can extract a witness with respect to the publicly-verifiable relation $\mathcal{R}_{\mathcal{G}}$.

Step 2 - public-verification of an image property. The family $\mathcal{G}$, after being augmented as above, still does not have a publicly-verifiable image property. We now construct from $\mathcal{G}$ a new family $\mathcal{F}$ that will already satisfy all the requirements of a verifiable robust family, as given by Definition 34.

The basic idea is to embed in the function's answers rZAPs (namely, ZAPs that are resettably witness indistinguishable, see Section 3.5.1) attesting that the answer is consistent with a given function determined by the verification key. However, to securely use rZAPs, we first need to establish a proper statement with at least two valid witnesses. We use a similar idea to that used in [58, 78] of using a WI proof (or rZAPs in our case) in order to get a witness-hiding proofs for statements with two independent witnesses. Details follow.

Each function f_k in the family $\mathcal{F}$ consists of two independent functions g_{k_0} and g_{k_1} sampled independently from $\mathcal{G}$, randomness $\mathbf{s}$ for an rZAP, and two random strings (r_0, r_1) ; the secret key k is set to be $(k_0, k_1, \mathbf{s})$. The corresponding verification key vk will consist of the two verification keys (vk_0, vk_1) sampled with the above to functions (as defined in Step 1), as well as two commitments $C_0 = \text{Com}(k_0)$, $C_1 = \text{Com}(k_1)$, and a third commitment $C = \text{Com}(b)$, to a random bit $b \leftarrow \{0, 1\}^n$ (the commitments are non-interactive and perfectly binding as defined in Section 3.5.1); overall, $vk = (vk_0, vk_1, C_0, C_1, C)$. The input sampler for $\mathcal{F}$ is given by the product $\hat{\mathcal{D}} = \mathcal{D} \times \mathcal{U}$; namely it consists of an input for the underlying (g_{k_0}, g_{k_1}) , and a random string r , which will be a uniformly random first message for the rZAP. The length of r is chosen so to guarantee a soundness error of at most $2^{-\Omega(n)}$.

The function f_k , given an input (q, r) , computes $a_0 = g_{k_0}(q)$ and $a_1 = g_{k_1}(q)$, and then treating r as the first message of an rZAP, and using the rZAP randomness $\mathbf{s}$, computes an rZAP proof for the statement

$$\psi(C_0, C_1, C, q, a_0, a_1) := \left\{ \{C = \text{Com}(0)\} \vee \left\{ \begin{array}{l} C_0 = \text{Com}(k_0; r_0) \\ a_0 = g_{k_0}(q) \end{array} \right\} \right\} \wedge \left\{ \{C = \text{Com}(1)\} \vee \left\{ \begin{array}{l} C_1 = \text{Com}(k_1; r_1) \\ a_1 = g_{k_1}(q) \end{array} \right\} \right\} ;$$

This statement has two witnesses: one witness consists of the bit b and randomness for the commitment C , as well as the key k_{1-b} , and randomness for the commitment C_{1-b} .

The second witness consists of the two keys (k_0, k_1) and the randomness corresponding to both commitments (C_0, C_1) . The function f_k correctly computes $a_b = g_{k_b}(q)$, for both $b \in \{0, 1\}$, and gives the proof using the second witness.

Finally, the unlearnable relation for the new family $\mathcal{F}$ will be:

$$\mathcal{R}_{\mathcal{F}} = \{((vk_0, vk_1, C_0, C_1, C), z) \mid z \in \mathcal{R}_{\mathcal{G}}(vk_0) \vee z \in \mathcal{R}_{\mathcal{G}}(vk_1)\} ;$$

namely, z is a witness if it is a witness for either one of the underlying verification keys vk_0 or vk_1 .

Claim 3.4.2. *The function family $\mathcal{F}$ is verifiable and robust according to Definition 34.*

Proof. First, we show that a random $f_k \in \mathcal{F}$ is black-box unlearnable, even given the verification key vk . Specifically, we show that any learner L that satisfies $\mathcal{R}_{\mathcal{F}}(vk)$ with noticeable probability $\varepsilon = \varepsilon(n)$ can be transformed into a learner L' for the family $\mathcal{G}$.

Given oracle access to a random g_k , and its verification key vk' , L' samples a random bit b on its own, and treats its oracle g_k as g_{k_b} and the verification key vk' as vk_b . In addition, L' samples $g_{k_{1-b}} \in \mathcal{G}$, together with a verification key vk_{1-b} . L' then feeds the learner L with a verification key vk consisting of commitments $C = \text{Com}(b), C_{1-b} = \text{Com}(k_{1-b})$ and $C_b = \text{Com}(0^{|k_b|})$, and the verification keys (vk_b, vk_{1-b}) . Now, L' emulates L , answering any query (q, r) using its own sampled function for the answer a_{1-b} , and the oracle for the answer a_b . The proof is then given using what we referred above as “the first witness”; namely, for the first part of the “and statement” b and the randomness for $C = \text{Com}(b)$ are used, and for the second part the function $g_{k_{1-b}}$ and the randomness for $C_{1-b} = \text{Com}(k_{1-b})$ is used.

We now claim that with probability at least $\varepsilon/2 - \text{negl}(n)$, the emulated L (and thus also L') outputs $z_b \in \mathcal{R}_{\mathcal{G}}(vk_b)$ for the oracle g_{k_b} . Indeed, using the fact that the commitment Com is hiding, we can first move to an hybrid experiment where C_b is a commitment to k_b , and then using the fact that the rZAP is (resetably) witness-indistinguishable, we can move to another hybrid experiment where “the second witness” for the WI statement is used. In this experiment L has the exact same view as in a true interaction with f_k , where it outputs z that satisfies the unlearnable relation corresponding to one of the two functions. Then, using again the hiding of the commitment, we can move to an experiment where C is a commitment to a bit b' that is chosen independently of b , and L would still succeed with probability $\varepsilon - \text{negl}(n)$. In

this experiment, the view of L is independent of b , and so it must output the right z_b at least in $\frac{1}{2}$ of its successes. The claim follows.

We now show that $\mathcal{F}$ is non-black-box learnable according to Definition 34. Specifically, we define the image relation $\text{Ver}_{\mathcal{F}}(vk, (q, r), (a_0, a_1, \pi))$ to be a relation that simply verifies the rZAP proof π with respect to a first random message r , and the statement given by $\psi(C_0, C_1, C, q, a_0, a_1)$, where (C_0, C_1, C) are determined by vk .

By the completeness of the rZAP, $\text{Ver}_{\mathcal{F}}$ has the required completeness: $\text{Ver}_{\mathcal{F}}(vk, (q, r), f_k(q, r)) = 1$ for any triple. Next, for $\varepsilon \geq 2^{-\sqrt{n}}$, let C be any circuit that $(\varepsilon, \widehat{\mathcal{D}})$ -approximates f_k in the weak sense given by Definition 34. The extractor $\mathcal{E}_{\mathcal{F}}$ will operate as follows: it will first transform C into two circuits C_0 and C_1 such that, with probability $1 - 2^{-\Omega(n)}$, one of them will $(\varepsilon, \nu, \mathcal{D})$ -approximate the underlying function g_{k_0} or g_{k_1} , for $\nu = 0$; namely, with no errors. The circuit C_b will have a random first rZAP message r hardwired into it. Given a sample q from $\mathcal{D}(1^n)$, C_b feeds (q, r) to C , when it gets a result (a_0, a_1, π) , it verifies the proof π ; if it does not verify, C_b outputs $\perp$, otherwise it outputs a_b .

Let b be the plaintext bit determined by the commitment C given in the verification key vk . By the soundness of the rZAP, with probability $1 - 2^{-\Omega(n)}$ over the choice of randomness r , C_b $(\varepsilon, 0, \mathcal{D})$ -approximates g_{k_b} . Hence, it suffices that the extractor $\mathcal{E}_{\mathcal{F}}$ feeds each one of (C_0, C_1) to the extractor $\mathcal{E}_{\mathcal{G}}$ of the family $\mathcal{G}$, and it will manage to produce $z \in \mathcal{R}_{\mathcal{G}}(k_b)$ with the required probability. $\square$

Remark 38 (Perfectly-binding commitments from trapdoor permutations). *In the above, we rely on non-interactive commitments. These are known based on trapdoor permutations (or even one-to-one one-way functions) that have a recognizable key space (namely, it is possible to verify that the given function is indeed injective), which is crucial to guarantee binding. In our application above, we need not assume explicitly such recognizability. Indeed, the circuit evaluating the unobfuscatable functions, in particular, provides a (statistical) ZAP proof for the validity of the commitment, which already ensures binding.*

This concludes the proof of Theorem 37 $\square$

3.5 From Verifiable Robust Unobfuscatable Functions to Resettable Protocols

In this section, we apply verifiable robust unobfuscatable functions in order to construct new resettable protocols.

3.5.1 Definitions

We briefly recall the definitions of resettable soundness [15, 129], resettable ZK [53], and several other basic definitions and tools used in this section. Most definitions are taken almost verbatim from [15, 53, 69]. In Subsection 3.5.1.1, we define instance-dependent resettable witness-indistinguishability.

Resettable-sound ZK We briefly recall the definitions of resettable soundness presented in [15]. In the setting of resettable soundness, the prover $\mathcal{P}^*$ has the power to reset the verifier $\mathcal{V}$. Specifically, the random tape of $\mathcal{V}$ is chosen at random and fixed once and for all and, from that point on, the prover can interact multiple times with the residual deterministic verifier $\mathcal{V}_r(x)$ induced by r and the common input x . Each such interaction is called a session.

Note that the adversary may repeat in a current session the same messages sent in a prior session, resulting in an identical prefix of an interaction (since the verifier's randomness is fixed). Furthermore, by deviating in the next message, the adversary may obtain two different continuations of the same prefix of an interaction.

A generalization of the above model, also considered in [15], is to allow the prover to interact with multiple “incarnations” of the verifier. Here $t = \text{poly}(n)$ random tapes $r_1, \dots, r_t$ are sampled, and the prover can adaptively choose at any point an input x and index $i \in [t]$ and interact with $\mathcal{V}_{r_i}(x)$.

Definition 39 (Resettable-sound argument [15]). *A resetting attack of a malicious prover $\mathcal{P}^*$ on a resettable verifier $\mathcal{V}$ is defined by the following random process, indexed by a security parameter n :*

1. *Uniformly select $t = \text{poly}(n)$ random-tapes, denoted $r_1, \dots, r_t$ for $\mathcal{V}$, resulting in deterministic strategies $\mathcal{V}_{r_1}, \dots, \mathcal{V}_{r_t}$. For a given $x \in \{0, 1\}^n$, $\mathcal{V}_{r_i}(x)$ is called an incarnation.*

2. A prover $\mathcal{P}^*$ of size $\text{poly}(n)$ can initiate $\text{poly}(n)$ interactions with different incarnations. In each such interaction, $\mathcal{P}^*$ chooses $x \in \{0,1\}^n$ and $i \in [t]$, and conducts a complete session with the incarnation $\mathcal{V}_{r_i}(x)$.

An argument system $(\mathcal{P}, \mathcal{V})$ is a *resettable-sound* argument for $\mathcal{L}$ if, for any resetting poly-size $\mathcal{P}^*$, the probability that in some session during a resetting attack, $\mathcal{P}^*$ convinces some incarnation $\mathcal{V}_{r_i}(x)$ of accepting while $x \notin \mathcal{L}$ is negligible in n . The system is also an argument of knowledge, if it is possible to extract a witness for $x \in \mathcal{L}$ from any resetting $\mathcal{P}^*$ that convinces $\mathcal{V}$ of accepting x in some session.

For simplicity, we concentrate on the simple case of one incarnation $\mathcal{V}_r(x)$. The results extend to the model of multiple incarnations using the transformation in [15]. At high-level, in this transformation, each r_i is a seed for a pseudo-random function, and the verifier applies the PRF to any input chosen by the prover to derive fresh randomness. In the case that the inputs in the different incarnations are chosen statically ahead of time, this transformation works smoothly for any resettable-sound argument by a reduction to the PRF security. However, for adaptive choice of inputs, the transformation only works for a resettable argument-of knowledge (the issue being that, without an argument of knowledge guarantee, it may be hard to tell whether the inputs chosen adaptively by the prover, are in the language). See further details in [15, 142].

Resettable-sound witness-indistinguishable arguments of knowledge. An *rsWIAOK*, is WI in the usual sense, and is also an argument of knowledge against resetting provers; namely, we can efficiently extract a witness from any resetting prover that convinces the verifier of accepting with noticeable probability. Such proof systems can be constructed from classical public-coin proof systems such as (the n -fold version of) Blum’s Hamiltonicity WI protocol [38] by applying to them the [15] transformation (the resulting constructions are based solely on one-functions).

A useful property of these classical protocols is that the knowledge extraction uses the prover as a black-box (except for being given its size).

Resettable ZK we start by recalling resettable ZK.

Definition 40 (Resettable zero-knowledge [53]). *An interactive proof system $(\mathcal{P}, \mathcal{V})$, for a language $\mathcal{L}$, is said to be resettable zero-knowledge if, for every poly-size adversary $\mathcal{V}^*$, there exists a probabilistic polynomial time simulator $\mathcal{S}$ so that the distribution*

ensembles $\mathcal{D}_1$ and $\mathcal{D}_2$ described below are computationally indistinguishable: let each distribution be indexed by a sequence of distinct common inputs $\vec{x} = x_1, \dots, x_{\text{poly}(n)} \in \mathcal{L} \cap \{0, 1\}^n$ and a corresponding sequence of prover's auxiliary-inputs $\vec{y} = y_1, \dots, y_{\text{poly}(n)}$.

Distribution $\mathcal{D}_1$ is defined by the following random process which depends on $\mathcal{P}$ and $\mathcal{V}^*$

- Randomly select and fix $t = \text{poly}(n)$ random-tapes $\omega_1, \dots, \omega_t$ for $\mathcal{P}$, resulting in deterministic strategies $\mathcal{P}^{(i,j)} = \mathcal{P}_{x_i, y_i, \omega_j}$ defined by $\mathcal{P}_{x_i, y_i, \omega_j}(\alpha) = \mathcal{P}(x_i, y_i, \omega_j, \alpha)$ for $i, j \in \{1, \dots, t\}$. Each $\mathcal{P}^{(i,j)}$ is called an incarnation of $\mathcal{P}$.
- Machine $\mathcal{V}^*$ is allowed to run polynomially-many sessions with the $\mathcal{P}^{(i,j)}$'s. Throughout these sessions, $\mathcal{V}^*$ is required to complete its current interaction with the current copy of $\mathcal{P}^{(i,j)}$ before starting a new interaction with any $\mathcal{P}^{(i',j')}$, regardless if $(i, j) = (i', j')$ or not. Thus, the activity of $\mathcal{V}^*$ proceeds in rounds. In each round it selects one of the $\mathcal{P}^{(i,j)}$'s and conducts a complete interaction with it.
- Once $\mathcal{V}^*$ decides it is done interacting with the $\mathcal{P}^{(i,j)}$'s it (i.e., $\mathcal{V}^*$) produces an output based on its view of these interactions. This output is denoted by $(\mathcal{P}(\vec{y}), \mathcal{V}^*)(\vec{x})$ and is the output of the distribution.

Distribution $\mathcal{D}_2$: The output of $\mathcal{S}(\mathcal{V}^*, \vec{x})$.

Again for simplicity, we concentrate on the simple case of one prover incarnation; however, all of our results directly extend to the model of multiple incarnations.

Resettable ZAPs. ZAPs are two-message public-coin witness-indistinguishable proofs introduced by Dwork and Naor [74]. They further have the special property that the first message (sent by the prover) can be reused for multiple proofs. As noted in [15], any ZAP already has the property of resettable soundness. Furthermore, resettable witness-indistinguishability property can be obtained by applying the transformation in [53]. We refer to the resulting system as an rZAP system having the property of resettable soundness as well as resettable-witness indistinguishability.

Commitments We shall use two types of perfectly (or statistically) binding commitments. One type is non-interactive commitments, which can be constructed based on

any one-way permutation (let alone a trapdoor permutation) [37]. The second are interactive (two-message) commitments that can be constructed from one-way functions [133].

3.5.1.1 Instance-dependent resettable witness-indistinguishable arguments

An instance-dependent resettable-WI argument (rWI^y), for an **NP** language $\mathcal{L}$, is defined with respect to a candidate instance y for another (possibly different) **NP** language $\mathcal{L}'$. In an instance-dependent rWI^y the soundness and rWI properties are decoupled according to y : if $y \in \mathcal{L}'$ the system is sound and if $y \notin \mathcal{L}'$ the system is rWI .

Definition 41 (rWI^y). *An argument system $(\mathcal{P}, \mathcal{V})$ for $\mathcal{L}$ is rWI with respect to $\mathcal{L}'$ if it satisfies:*

1. **Instance-dependent soundness:** *for any poly-size prover $\mathcal{P}^*$, any $x \in \{0, 1\}^n \setminus \mathcal{L}$, and $y \in \mathcal{L}'$:*

$$\Pr_{\mathcal{V}} [(\mathcal{P}^*, \mathcal{V})(x, y) = 1] \leq \text{negl}(n) .$$

2. **Instance-dependent rWI :** *for any poly-size resetting verifier $\mathcal{V}^*$, any $x \in \mathcal{L}$, $w_x^{(1)}, w_x^{(2)} \in \mathcal{R}_{\mathcal{L}}(x)$, and $y \notin \mathcal{L}'$:*

$$\Pr \left[\left(\mathcal{P}(w_x^{(1)}), \mathcal{V}^* \right) (x, y) = 1 \right] - \Pr \left[\left(\mathcal{P}(w_x^{(2)}), \mathcal{V}^* \right) (x, y) = 1 \right] \leq \text{negl}(n) .$$

Constructing instance-dependent rWIs from one-way functions Instance-dependent rWI can be constructed based on the Canetti-Goldreich-Goldwasser-Micali transformation [53], and instance-dependent commitment schemes (also known as trapdoor commitments or chameleon commitments), which in turn can be constructed from one-way functions [77].

We briefly recall what are the properties of such commitments (a more formal definition can be found in [77]). An instance-dependent commitment scheme is also parameterized by an instance y for an **NP** language $\mathcal{L}'$ and has two properties:

1. **Instance-dependent equivocation:** There is an efficient equivocation algorithm Eq that, given $(y, w) \in \mathcal{R}_{\mathcal{L}'}$ and some length ℓ , can produce an equivocal

commitment $\tilde{C}$, which it can open to any $x \in \{0, 1\}^\ell$ (the equivocation algorithm uses w). Equivocal commitments and their decommitment information are computationally indistinguishable from honestly generated commitments and their decommitment information.

- 2. Instance-dependent binding:** For $y \notin \mathcal{L}$, the commitment is statistically binding.

Given such commitments, it is possible to transform any public-coin WI system (e.g. Blum [38]) to an instance-dependent rWI, using the [53] transformation. Specifically, given an instance y as a candidate for $\mathcal{L}$, the verifier first commits to its random message using the y -dependent commitment. Then, the parties run the WI protocol, where the verifier opens the commitment as its message. The randomness used by the prover is derived by applying a pseudo random function to the verifier's commitment. If $y \in \mathcal{L}'$, it follows by the equivocation guarantee, and the soundness for the underlying WI, that the protocol is sound. If $y \notin \mathcal{L}$, the commitment is binding, and rWI follows as in [53].

3.5.2 A Basic Residual Probability Bound

The following probabilistic claim will be used more than once in this section. The proof of the claim is standard and appears in Section 2.3.

Let $S_1, \dots, S_m$ be random variables (intuitively, describing an m -step process), and let $G_1, \dots, G_m$ be events where G_i is determined by S_i alone. Denote by H_i the event that the first $i - 1$ stages $S_1, \dots, S_{i-1}$ are such that the event G_i occurs with bounded probability δ (when S_i is sampled conditioned on $S_1, \dots, S_{i-1}$); namely, H_i is the event that $S_1, \dots, S_{i-1}$ satisfy:

$$\Pr_{S_i | S_1, \dots, S_{i-1}} [G_i \mid S_1, \dots, S_{i-1}] \leq \delta. \quad ^3$$

The following intuitive claim shows an exponential decay in the probability that k events G_i occur when the residual probability of their occurrence is bounded through every stage of the process.

³More explicitly, we mean that H_i is that even that $(S_1, \dots, S_{i-1}) = (s_1, \dots, s_{i-1})$, such that $(s_1, \dots, s_{i-1})$ satisfy $\Pr_{S_i | (S_1, \dots, S_{i-1}) = (s_1, \dots, s_{i-1})} [G_i \mid (S_1, \dots, S_{i-1}) = (s_1, \dots, s_{i-1})] \leq \delta$.

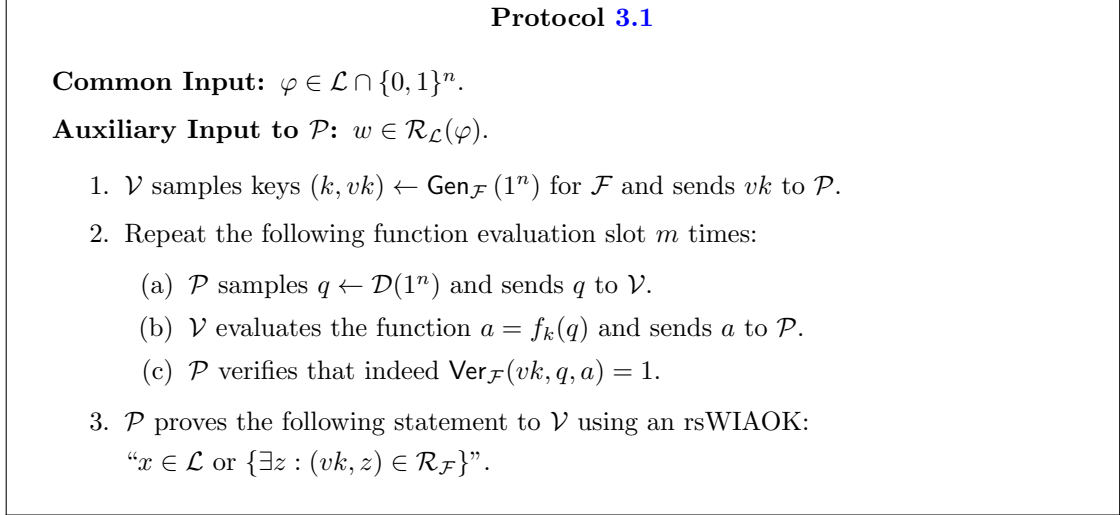


Figure 3.1: A resettably-sound (concurrent) ZK protocol

Claim 3.5.1. *For every $\{i_1, i_2, \dots, i_k\} \subseteq [m]$:*

$$\Pr_{S_1, \dots, S_m} \left[\left(\bigwedge_{j=1}^k G_{i_j} \right) \wedge \left(\bigwedge_{j=1}^m H_j \right) \right] \leq \delta^k .$$

3.5.3 The Base Protocol

In this section, we present an $O(m)$ -round resettably-sound ZK protocol where m is a parameter that will be chosen according to the desired notion of ZK (stand-alone or concurrent).

In what follows, let $\mathcal{F}$ be a verifiable robust unobfuscatable function family, with respect to the relation $\mathcal{R}_{\mathcal{F}}$ and input distribution $\mathcal{D}$. Let $\text{Gen}_{\mathcal{F}}, \text{Ver}_{\mathcal{F}}$ be the key generation algorithm and the image verification relation of $\mathcal{F}$. The protocol is detailed in Figure 3.1.

3.5.4 Security Analysis

In this section, we analyze the resettable security of Protocol 3.1, according to the different settings of the parameter m .

3.5.4.1 Resettable Soundness

We show that Protocol 3.1 is resettably-sound, for any setting of m (constant or $\text{poly}(n)$).

Lemma 42. *For any m , Protocol 3.1 is a resettably-sound ZK argument of knowledge.*

Proof. Let $\mathcal{P}^*$ be a poly-size prover that, for a set $\mathcal{X}$ of inputs, convinces $\mathcal{V}$ to accept each $x \in \mathcal{X} \cap \{0,1\}^n$ with probability $\varepsilon(n)$. We describe a PPT extractor $\mathcal{E}$ that, given black-box access to $\mathcal{P}^*$ extracts a witness $w \in \mathcal{R}_{\mathcal{L}}(x)$, for all but finitely many $x \in \mathcal{X}$. First, we consider a new prover $\mathcal{P}_1^*$ against the rsWIAOK protocol concluding the proof. The new prover $\mathcal{P}_1^*$ is given oracle access to a random $f_k \in \mathcal{F}$, as well as a corresponding public key vk , and tries to convince the rsWIAOK verifier of the WI statement corresponding to x and vk . This prover perfectly emulates $\mathcal{P}^*$, forwarding to the oracle f_k any query that $\mathcal{P}^*$ makes in the function evaluation slot, and forwarding any message in the proof stage to the external resettable verifier. Since the view of the emulated $\mathcal{P}^*$ is the same as in a real execution, it convinces the rsWIAOK verifier with the same probability ε .

We can now apply the black-box extractor $\mathcal{E}_{\text{wi}}$ for the rsWIAOK to extract a witness from $\mathcal{P}_1^*$. To do so, $\mathcal{E}_{\text{wi}}$ can sample f_k and vk on its own, and use them to answer any oracle call that $\mathcal{P}_1^*$ makes. By the POK guarantee, we know that $\mathcal{E}_{\text{wi}}$ outputs a witness for the WI statement corresponding to x and vk (in expected polynomial time). It is left to see that, except with negligible probability, this witness will be $w \in \mathcal{R}_{\mathcal{L}}(x)$, rather than $z \in \mathcal{R}_{\mathcal{F}}(vk)$; otherwise, we can use $\mathcal{E}_{\text{wi}}$ and $\mathcal{P}_1^*$ to break the unlearnability of $\mathcal{F}$. Indeed, since $\mathcal{E}_{\text{wi}}$ is a black-box extractor (see Section 3.5.1), it can answer all of $\mathcal{P}_1^*$'s calls to the function, using an external oracle to f_k . $\square$

3.5.4.2 Stand-Alone (Non-Resetable) ZK

In this section, we show that Protocol 3.1 for $m = O(1)$ is (stand-alone) ZK. In the next section, we'll show that for $m = n^{\Omega(1)}$, it is concurrent ZK.

Let $\mathcal{F}$ be a verifiable robust unobfuscatable function family, as the one used in Protocol 3.1, and let d be a constant such that the running time of the non-black-box extractor of $\mathcal{F}$, given security parameter n , and approximation parameter ε , is bounded by $\text{poly}(n, |C|) \cdot \varepsilon^{-d}$. We will show that the protocol is resettably-sound when $m \geq d$.

At high-level, the correlation between d and m is required to ensure that our simulator would be able to run in expected poly-time. Here the problematic case is when the probability $\varepsilon(n)$ that the verifier finishes the execution is neither noticeable or negligible. In such a case, to make sure that the simulated distribution is not skewed, the simulator should complete the execution with roughly the same probability $\varepsilon = \varepsilon(n)$ as

the verifier. The problem is that in such a case naively trying to extract from a single slot would take time $\text{poly}(n, |\mathcal{V}^*|) \cdot \varepsilon^{-d}$, so overall the simulator's running time would be roughly $\text{poly}(n, |\mathcal{V}^*|) \cdot \varepsilon^{-d+1}$, which may not be polynomially bounded.

The addition of extra slots allows to ensure that with high-probability in one of these slots the probability that the verifier finishes the slot is roughly $\varepsilon^{\frac{1}{d}}$. Our simulator can thus first estimate the probability ε , and then make repeated trials to find a circuit that $\varepsilon^{\frac{1}{d}}$ approximates the unobfuscatable function and extract the trapdoor from this slot.

Lemma 43. *Protocol 3.1 with $m \geq d$ is ZK.*

Proof. Similarly to the simulator of Goldreich and Kahan [97], our simulator starts by running the cheating verifier $\mathcal{V}^*$ once, honestly simulating its view until the proof step. We say that $\mathcal{V}^*$ aborts if in some slot it does not answer or returns an answer a to a query q that does not verify under the verification key vk it sent. If $\mathcal{V}^*$ does not abort and evaluates the function correctly in each one of the d slots, the simulator interacts with $\mathcal{V}^*$ repeatedly to estimate the probability ε that $\mathcal{V}^*$ evaluates the function correctly in all the slots. Specifically it samples executions of the protocol until it obtains n successes and then sets the estimate as $\tilde{\varepsilon} = n/N$, where N is the number of trials.

Using a standard tail bound, it can be seen that, except with probability $2^{-\Omega(n)}$, it holds that $\tilde{\varepsilon} \in (\frac{\varepsilon}{2}, 2\varepsilon)$; throughout the rest of the proof, we assume that this is indeed the case.

The next step of the simulator is to obtain a slot that computes the function f_k with high-enough probability. Let $S_1, \dots, S_d$ be random variables describing each of the d slots, and let G_i be the event that the verifier does not abort in the i -th slot, and let $\neg H_i$ be the event that $S_1, \dots, S_{i-1}$ are such that the verifier does not abort in S_i with probability at least $(\varepsilon/2)^{1/d}$, i.e. $\Pr[G_i \mid S_1, \dots, S_{i-1}] \geq (\varepsilon/2)^{1/d}$. The simulator is going to try and sample $S_1, \dots, S_{i-1}$ such that $\neg H_i$ holds and then use the residual circuit for the slot S_i to extract the key. To do so, our simulator will repeatedly sample protocol executions until it obtains n independent protocol executions $S_1, \dots, S_d$ that are non aborting; namely, such that $G_1, \dots, G_d$ hold. For each of these n executions, the simulator would create d residual circuits, for each one of the slots, and run the extractor of $\mathcal{F}$, with approximation parameter $(\tilde{\varepsilon}/4)^{1/d}$, provided that $(\tilde{\varepsilon}/4)^{1/d} \geq 2^{-\sqrt{n}}$; if this is not the case, the simulator aborts, which has a negligible $2^{-\Omega(\sqrt{n})}$ statistical effect on the simulated distribution.

We claim that, except with probability $2^{-\Omega(n)}$, the simulator does not “get stuck”; that is, it obtains the trapdoor from one of the $n \cdot d$ circuits. Correctness of the simulation then follows from the WI property of rsWIAOK. We now prove this, and then analyze the running time of the simulator.

First, we claim that conditioned on sampling a non-aborting execution $S_1, \dots, S_d$, the event $\neg H_i$ occurs for some i , with probability at least $\frac{1}{2}$ over the choice of $S_i | S_1, \dots, S_{i-1}$. Indeed, by the residual probability Claim 3.5.1, it holds that $\Pr[G_1, \dots, G_d, H_1, \dots, H_d] \leq \left((\varepsilon/2)^{1/d}\right)^d \leq \varepsilon/2$, when setting $k = m = d$, and $\delta = (\varepsilon/2)^{1/d}$, and thus

$$\begin{aligned} \Pr[\exists i : \neg H_i \mid G_1, \dots, G_d] &= \\ 1 - \Pr[H_1, \dots, H_d \mid G_1, \dots, G_d] &= \\ 1 - \frac{\Pr[G_1, \dots, G_d, H_1, \dots, H_d]}{\Pr[G_1, \dots, G_d]} &\geq \\ 1 - \frac{\varepsilon/2}{\varepsilon} &. \end{aligned}$$

It follows that, except with probability 2^{-n} , the simulator indeed finds a circuit satisfying $\neg H_i$. When it runs the extractor with approximation parameter $(\tilde{\varepsilon}/4)^{1/d} \leq (\varepsilon/2)^{1/d}$, it will manage to extract the key k with probability $1 - 2^{-\Omega(n)} \cdot |\mathcal{V}^*|^{O(1)}$.

It is left to analyze the simulator’s running time. The simulator simulates an execution only if the initial transcripts was accepting, which happens with probability ε . Hence, it suffices to show that the expected running time of the rest of the simulation is bounded by $\text{poly}(n)/\varepsilon$. Indeed, estimating $\tilde{\varepsilon}$ takes expected time n/ε . Then rejection sampling for each non-aborting execution $S_1, \dots, S_d$ takes time $1/\varepsilon$, and overall n/ε . Then, for each of the $n \cdot d$ circuits, the extractor runs in expected time $\text{poly}(n, |\mathcal{V}^*|) \cdot (\varepsilon^{1/d})^{-d}$. Thus, the overall running time is bounded as required by $\varepsilon \cdot \text{poly}(n)/\varepsilon = \text{poly}(n)$. $\square$

Round-efficient resettable-sound ZK As explained in Section 3.3.5, the construction described in Section 3.3 either yields extractors with running time $\text{poly}(n) \cdot (|C|/\varepsilon)^2$, assuming trapdoor permutations, or extractors running in time $\text{poly}(n) \cdot |C|/\varepsilon$, assuming fully-homomorphic encryption. Plugging this into Protocol 3.1, we would get by Lemma 43, an eight-message protocol in the first case and a six-message one in the second case, if we use say a three-message rsWIAOK (e.g. based on [38]). However, we

can, in fact, save a round and run the first two messages of the rsWIAOK in parallel to the last slot of the protocol (Step 2) similarly to the protocol of [77].

Corollary 44 (of Lemma 43). *Assuming trapdoor permutations there exists a six-message resettably-sound ZK protocol.*

Corollary 45 (of Lemma 43). *Assuming fully homomorphic encryption and trapdoor permutations, there exists a four-message resettably-sound ZK protocol.*

3.5.4.3 Concurrent and Resettable ZK

In this section, we show that for $m = n^{\Omega(1)}$ Protocol 3.1 is concurrent ZK. A simultaneously-resettable ZK protocol can then be obtained from Protocol 3.1 and any simultaneously-resettable WI protocol by applying a general transformation of Deng, Goyal, and Sahai [69].

Lemma 46. *For every constant $\delta > 0$, Protocol 3.1 with $m = n^\delta$ is concurrent zero-knowledge.*

Proof. We start with an overview of the simulator.

Overview of the simulation The simulation uses the techniques of Richardson and Kilian [146], and more specifically, the slightly augmented version of them applied in [55, 69]. We start by describing the main ideas behind this technique as they were used in previous works. The execution of the protocol in every session consists of running many sequential slots giving the simulator many chances to “solve” the session. The simulation runs a *main thread*, and in the beginning of every slot, the simulation also starts “look-ahead” threads that are only meant to simulate the interaction until the end of the slot. If the simulation in the look-ahead thread is successful, the simulator can then continue the simulation in the main thread and solve the session.

The main difficulty is that, in the concurrent setting, even the simulation of one slot may be non trivial. The idea is to have look ahead threads recursively use the same simulation strategy. If some slot contains too many other concurrent slots, the simulation of the corresponding look-ahead thread may “give up” and not reach the end of the slot. However, it can be shown that, in every session, some slots must be successfully simulated by the look-ahead threads and therefore in the main thread, all sessions will be solved.

Our setting In our simulation, we think of every slot as computing $\mathcal{V}^*$'s function f_k . Instead of actually running a look-ahead thread to simulate the execution of a slot, we construct the circuit implementing such look-ahead thread and run the extractor of the unobfuscatable function family on this circuit. While we do not start any look ahead threads, the nature of our simulation is still recursive since the circuits we construct and extract from construct smaller simulation circuits themselves and extract from them.

We now describe the basic sub-routines used in our simulation procedure. We first describe them at high-level, and then move on to describe them in more detail.

Let $\mathcal{V}^*$ be a concurrent verifier that opens at most n^c sessions, and assume WLOG that $\mathcal{V}^*$ is deterministic. We refer to every iteration of Step 2 where $\mathcal{V}^*$ evaluates the function f_k as a slot. In a single session, there are n^δ slots, and thus in the entire execution of $\mathcal{V}^*$ with the honest prover, there are at most $n^{c'} = n^{c+\delta}$ slots (less if $\mathcal{V}^*$ aborts or deviates from the protocol). The simulator $\mathcal{S}$, using the code of $\mathcal{V}^*$, uses the following functions to create the simulated transcript:

The function MAIN simulates a given number of slots. MAIN takes the parameters (T, m) , where T is the state of the concurrent simulation at the time the function is called, and m is the number of additional slots to simulate.

The function EMULATE transforms the code of $\mathcal{V}^*$ into a circuit that computes the function f_k . EMULATE is given a state T of a partial simulation of $\mathcal{V}^*$, right before the beginning of a slot. EMULATE generates a circuit $\mathcal{C}$ that, given input q , tries to evaluate the function $f_k(q)$ by simulating the slot with the input q as the first message. $\mathcal{C}$ simulates $\mathcal{V}^*$ according to same logic as in the function MAIN, until the slot terminates. EMULATE takes the parameters (T, m) , and uses them within $\mathcal{C}$ in the same way that they are used in the MAIN function.

The function EXTRACT tries to extract the unlearnable secret z corresponding to the function f_k , in some session, by executing the extractor $\mathcal{E}$ of the unobfuscatable function family on the circuit $\mathcal{C}$ returned by EMULATE. EXTRACT takes parameters (T, m) and passes them to EMULATE.

We note following recursive relations between the above functions:

- When $m > 1$ the function MAIN will execute the function EXTRACT on every slot it starts, passing it a smaller m .
- The function EXTRACT uses the circuit $\mathcal{C}$ generated the function EMULATE.

- The circuit $\mathcal{C}$ generated by the function EMULATE simulates messages according to the strategy of the function MAIN. In particular, $\mathcal{C}$ executes the function EXTRACT with a smaller value of m .

The initial call The simulator $\mathcal{S}$ will execute the function MAIN with $m = n^{c'}$ and the empty simulation state.

Next, we describe the above functions in more detail.

The function MAIN The function is given the state T of the simulation so far, and extends it as described above, updating the value of T as the simulation proceeds. Specifically, the simulation up to Step 3 (sending the rsWIAOK) in all sessions follows the strategy of the honest prover $\mathcal{P}$ (note that the witness w is not required for that). The state of the simulation is just the state maintained by $\mathcal{V}^*$ along the different sessions.

Whenever a slot starts, MAIN executes the function EXTRACT with parameters (T', m') where T' is the simulation state just before the beginning of the slot, and $m' = m/\Delta$ where $\Delta = n^{\delta'}$ for some $0 < \delta' < \delta$ a parameter of the simulation. If EXTRACT manages to output an unlearnable secret z such that $\mathcal{R}(vk, z) = 1$ for the vk sent in the session, then z is saved in T .

To simulate the rsWIAOK given in some session (Step 3), MAIN uses the unlearnable secret of the session saved in T . If no such secret is previously extracted MAIN aborts. In this case, we say that the simulation “got stuck”, and main will return $\perp$. When $\mathcal{V}^*$ terminates or starts the $(m + 1)$ -st slot, MAIN terminates and outputs the current value of T .

The function EMULATE On parameters (T, m) , the function creates a circuit $\mathcal{C}$ and outputs it. $\mathcal{C}$ simulates $\mathcal{V}^*$ by following almost the same logic as in the function MAIN with the same parameters (T, m) . The only difference between the simulation of $\mathcal{C}$ and the function MAIN is that $\mathcal{C}$ gets an external input q and sends it as the prover message in the first slot that $\mathcal{V}^*$ starts (when executed from T). If during the simulation, the first slot ends, $\mathcal{C}$ obtains $\mathcal{V}^*$ ’s response a and outputs it; otherwise, it outputs $\perp$. As in the function MAIN, if $\mathcal{V}^*$ aborts or starts m additional slots, before the first slot terminates, $\mathcal{C}$ outputs $\perp$.

The function EXTRACT On parameters (T, m) , the function calls the function EMULATE with the same parameters and obtains a circuit $\mathcal{C}$. EXTRACT then executes the extractor $\mathcal{E}$ of the unobfuscatable function family on $\mathcal{C}$ with threshold $\varepsilon = \frac{1}{16}$ (see Definition 34), and outputs the same as $\mathcal{E}$.

Correctness of the simulation Let $\text{Real}(\varphi, w)$ be the view of $\mathcal{V}^*$ in a real interaction $(\mathcal{P}(w), \mathcal{V}^*)(\varphi)$, and let $\mathcal{S}(\varphi)$ be the view generated by our simulation procedure. Let Bad be the event, over the coins of $\mathcal{S}$, that the simulation gets stuck; namely, it reaches a proof in some session in which the unlearnable secret was not yet extracted. Since the only difference between the experiments Real and $\mathcal{S}$ is the witness used to simulate the proof, it follows from the WI property of rsWIAOK that

$$\{\text{Real}(\varphi, w)\}_{(\varphi, w) \in \mathcal{R}_{\mathcal{L}}} \approx_c \{\mathcal{S}(\varphi) | \neg \text{Bad}\}_{\varphi \in \mathcal{L}} .$$

It is thus enough to show that

$$\Pr_{\mathcal{S}(\varphi)} [\text{Bad}] = \text{negl}(n) ,$$

where $n = |\varphi|$. For this purpose, we need to show that, in every session, one of the extraction attempts succeeds. The difficulty is that the circuit we are extracting from is performing simulation recursively, and this simulation may also get stuck and result in one of circuits (created by EMULATE along the main thread) not computing the function correctly. To bound the probability of such an event, we consider the following mental experiment: during simulation, before executing the extractor on some circuit, we execute this circuit several times (using independent randomness) to see if executing this circuit is likely to get stuck. The simulation in the circuit will recursively follow the same augmented behavior. If in any of the executions the circuit gets stuck, we do not even try to extract, but just admit failure. We show that the probability of getting stuck, in this modified experiment, is still negligible. Intuitively, this guarantees that no attempt to extract, in the real simulation, will ever fail due to a recursive simulation getting stuck (except with negligible probability). We next define this experiment in more detail.

Let $\mathcal{S}'$ be the experiment that is defined like $\mathcal{S}$, except that all the calls to the function EXTRACT are replaced by calls to a new function FORK. The function

FORK starts off just like the function EXTRACT; that is, on parameters (T, m) , FORK calls the function EMULATE with the same parameters, obtains a circuit $\mathcal{C}$, and executes the extractor $\mathcal{E}$ of the unobfuscatable function family on $\mathcal{C}$. FORK then repeats the following n independent times: sample $q \leftarrow \mathcal{D}(1^n)$ and run $\mathcal{C}(q)$. We shall refer to the computation transcript produced by $\mathcal{C}$ in the i -th execution (out of n executions) as the i -th simulation performed by FORK.

After every such simulation performed by FORK, the simulation is rewound back to the state T . If any of the simulations by FORK gets stuck, we say that FORK gets stuck as well, and this failure propagates up. More precisely:

- If the function MAIN makes a call to FORK that gets stuck, the entire simulation gets stuck.
- If FORK makes a recursive call to FORK that gets stuck, the outer execution of FORK also gets stuck.
- If a circuit $\mathcal{C}$ constructed by EMULATE makes a call to FORK that gets stuck, $\mathcal{C}$ outputs $\perp$.

Finally, FORK outputs the same as $\mathcal{E}$ on $\mathcal{C}$ (similarly to the function EXTRACT).

Clearly

$$\Pr_{\mathcal{S}'(\varphi)}[\text{Bad}] \geq \Pr_{\mathcal{S}(\varphi)}[\text{Bad}] ,$$

and therefore it is enough to show that

$$\Pr_{\mathcal{S}'(\varphi)}[\text{Bad}] = \text{negl}(n) .$$

Assume towards contradiction that there exists a polynomial p such that, for infinitely many values of $\varphi \in \mathcal{L} \cap \{0, 1\}^n$, $\Pr_{\mathcal{S}'(\varphi)}[\text{Bad}] \geq p(n)$, and let us fix any such φ . Whenever **Bad** occurs in $\mathcal{S}'$, the simulation performed by MAIN or by some call to FORK reaches the proof in some session while the unlearnable secret was not previously extracted. We look at the execution “thread” that contains the entire simulated interaction from the beginning of the experiment up until the point where the simulation gets stuck. (Note that this thread might spread across several nested calls to FORK.) We can identify every thread of execution by the number of previous calls to

FORK and the index of the simulation in the current call to FORK for every level of the recursion.

Let $\text{Bad}_1^{t,s}$ be the event where the execution gets stuck in thread t and session s . Since the total number of threads and sessions is polynomial, there exist a thread t and a session s such that the event $\text{Bad}_1 = \text{Bad}_1^{t,s}$ occurs with some polynomial probability $p_1(n)$.

To argue that some of the slots of session s in thread t are “light”, that is, they do not contain too many slots of other sessions, we will focus on a single level of the recursion that contains many of the slots of the session. Let Bad_2^l be the event that Bad_1 occurs and the call to FORK, in recursion level l and thread t , contains at least 2Δ full slots of session s . Since the simulation in the thread t must contains all n^δ slots of session s , in order to get stuck in s , and since the maximal number of nested calls to FORK is bounded by $d = \frac{c'}{\delta'} = O(1)$, it follows that there exist a level l such that the event $\text{Bad}_2 = \text{Bad}_2^l$ occurs with some polynomial probability $p_2(n)$. Let m_l be the parameter of FORK in recursion level l .

Since the simulation of FORK in thread t in level l of the recursion contains a bounded amount of slots from all sessions, but many slots of session s , there must be many slots of session s that are concurrent to a relatively small number of other slots. Intuitively, this means that the extraction from these light slots is more likely to succeed. Let S_i be the random variable representing the simulation of the i -th slot of session s that starts on recursion level l , in thread t (if no such slot exist S_i is empty). Let G_i be the event that S_i is not empty, contains no aborts, and the number of other slots that start concurrently to S_i is at most $\frac{m_l}{\Delta}$. Recall that before the simulation of a slot on recursion level l , $\mathcal{S}'$ makes a recursive call to FORK with parameter $\frac{m_l}{\Delta}$. Since the total number of sessions that start in the function FORK in level l is at most m_l , we have that whenever Bad_2 occurs, $|\{i \in [2\Delta] | G_i\}| \geq \Delta$.

Since with noticeable probability level l of thread t has many light slots, we expect that in at least one of these slots, if we rewind the simulation to the beginning of the slot, and simulate it again with independent randomness it will remain light with some constant probability. We will show that the extraction from such slot is likely to succeed. Let H_i be the event that $\Pr[G_i | S_1, \dots, S_{i-1}] < 1/8$. Let Bad_3^i be the event that $\text{Bad}_2 \wedge \neg H_i$. The following lemma, together with the fact that $\Pr_{\mathcal{S}'(\varphi)}[\text{Bad}_2] = p_2(n)$, implies that there exist $i^* \in [2\Delta]$ such that event $\text{Bad}_3 = \text{Bad}_3^{i^*}$ occurs with some

polynomial probability $p_3(n)$.

Lemma 47. $\Pr[\{|\{i \in [2\Delta] | G_i\}| \geq \Delta\} \wedge \{H_1 \wedge \dots \wedge H_{2\Delta}\}] \leq \text{negl}(n)$

Proof. By the residual probability Claim 3.5.1, when setting $m = 2\Delta, k = \Delta, \delta = \frac{1}{8}$, and fixing any set $\{i_j\}_{j \in [\Delta]} \subset [2\Delta]$, we have:

$$\Pr \left[\left(\bigwedge_{j=1}^{\Delta} G_{i_j} \right) \wedge \left(\bigwedge_{j=1}^{2\Delta} H_j \right) \right] \leq 8^{-\Delta} .$$

Furthermore, there are $\binom{2\Delta}{\Delta} < 2^{2\Delta}$ sets $\{i_j\}_{j \in [\Delta]} \subseteq [2\Delta]$ of size Δ and therefore:

$$\Pr [\{|\{i \in [2\Delta] | G_i\}| \geq \Delta\} \wedge \{H_1 \wedge \dots \wedge H_{2\Delta}\}] \leq \left(\frac{4}{8} \right)^{\Delta} \leq \text{negl}(n) .$$

□

Let T be the state of the simulation just before the beginning of the i^* -th slot. Let **good** be the event that the i^* -th slot exists, and the probability over the rest of the experiment, starting from T , that **Bad**₃ occurs is at least $p_3(n)/2$. Since $\Pr_{\mathcal{S}'(\varphi)}[\mathbf{Bad}_3] = p_3(n)$, we have that $\Pr_{\mathcal{S}'(\varphi)}[\mathbf{good}] \geq p_3(n)/2$. Now, conditioned on a **good** prefix T of the execution, $\Pr_{S_i} [G_i \mid T] \geq 1/8 = 2/16$ (this will be useful in a few lines).

Before the simulation of the i^* -th slot the function **FORK** is called, and tries to extract the unlearnable secret of the session s . Let S'_{i^*} be a random variable representing a simulation attempt made by **FORK** (out of the n identically distributed attempts). S'_{i^*} is distributed like S_{i^*} except that:

- S'_{i^*} never contains more than $\frac{m_l}{\Delta}$ other slots, while S_i may contain up to m_l concurrent slots. However, this difference does not hold conditioned on the event G_{i^*} .
- The simulation of S'_{i^*} makes recursive calls to **FORK** with parameter $\frac{m_l}{\Delta^2}$, while the simulation of S_{i^*} calls **FORK** with parameter $\frac{m_l}{\Delta}$; this means that the simulation of S_{i^*} is more likely to extract a witness, and less likely to get stuck. In particular, note that the above difference no longer holds conditioned on the event that the simulation of S'_{i^*} does not get stuck. We denote this event by **NS**.

Since G_{i^*} implies that S_{i^*} contains the end of the i^* -th slot, and that the messages (q, a) of the i^* -th slot satisfy $\text{Ver}_{\mathcal{F}}(vk, q, a) = 1$, we have that $\mathbf{NS} \wedge G_{i^*}$ implies that the

same holds for S'_{i^*} . Note that **NS** must occur with probability at least $15/16$; otherwise, one of the n simulation attempts made by **FORK**, starting from T , will get stuck with overwhelming probability and therefore **Bad**₂ will only occur with negligible probability; however, this will contradict our choice of T (recall that **Bad**₂ occurs only when the simulation gets stuck in thread t and not in the recursive call to **FORK** after state T). We thus have that $\Pr[\mathbf{NS} \wedge G_{i^*} \mid T] \geq 1/16$ and therefore, with probability at least $1/16$, S'_{i^*} contains the end of the i^* -th slot, and an execution of $\mathcal{C}(q)$, performed by **FORK** starting from state T , outputs a such that $\text{Ver}_{\mathcal{F}}(vk, q, a) = 1$ with probability at least $1/16$. By the non-black-box learnability property of the unobfuscatable function, when **FORK** invokes $\mathcal{E}$ on $\mathcal{C}$ with $\varepsilon = \frac{1}{16}$, it outputs z such that $(vk, z) \in \mathcal{R}_{\mathcal{F}}$ with overwhelming probability. However, by the choice of T , the probability that **Bad**₃ occurs and session s is not solved on thread t must be noticeable.

Simulation running time Let $T_{\text{MAIN}}(m), T_{\text{EMULATE}}(m), T_{\text{EXTRACT}}(m)$ be the running times of the functions **MAIN**, **EMULATE**, **EXTRACT**, given parameter m . One can verify that following relations hold:

$$\begin{aligned} T_{\text{MAIN}}(m) &< p_1(n) \cdot T_{\text{EXTRACT}}\left(\frac{m}{\Delta}\right) + p_2(n) \\ T_{\text{EXTRACT}}(m) &< p_3(T_{\text{EMULATE}}(m)) \\ T_{\text{EMULATE}}(m) &< p_4(T_{\text{MAIN}}(m)) , \end{aligned}$$

and therefore

$$T_{\text{MAIN}}(m) < p_1(n) \cdot p_5\left(T_{\text{MAIN}}\left(\frac{m}{\Delta}\right)\right) + p_2(n) ,$$

where p_1 to p_5 are polynomials that depend only on $\mathcal{V}^*$. Specifically, note that p_3 is such a polynomial since **EXTRACT** only runs $\mathcal{E}$ with a constant value of ε . For $m \leq 1$ the function **MAIN** stops whenever a new slot starts and does not make any calls to **EXTRACT**. Therefore, $T_{\text{MAIN}}(m) < p_0(n)$ where p_0 is a polynomial that depend only on $\mathcal{V}^*$. We get that:

$$T_{\text{MAIN}}(m) < p^d(n)$$

where p is a polynomial that depends only on $\mathcal{V}^*$ and $d = \lceil \log_{\Delta} m \rceil + 1$. Since $\Delta = n^{\delta'}$ and the main simulation executes MAIN with parameter $m = n^{c'}$ we have $d = O(1)$ and the simulation running time is polynomial.

This concludes the proof of Lemma 46 showing that Protocol 3.1 is concurrent zero-knowledge. □

3.5.5 Resettable Protocols from Minimal Assumptions

In this section, we show how to construct resettable-sound ZK, and resettable-sound concurrent ZK protocols based only on one-way functions, by directly using robust unobfuscatable function, instead of (ZAP-based) verifiable robust ones. As a first corollary, by plugging our protocol into the transformation of [15], we get an $\tilde{O}(\log n)$ -round resettable ZK argument of knowledge based on one-way functions. As a second corollary, when combining our protocol with the simultaneously-resettable WI from one-way functions [61], and a general transformation of [69], we obtain a *simultaneously-resettable ZK protocol from one-way functions*.

We now proceed to describe the augmented protocol. Let $\mathcal{G}$ be a family of robust unobfuscatable functions with respect to a hardcore family $\{\mathcal{HC}_n\}$ (as given by Definition 21, and constructed in Sections 3.2 and 3.3 based on one-way functions). Let φ be a one-way function. We make use of a two-message statistically binding commitment Com (e.g., [133]). Given a first receiver message r , we denote by Com_r the function that computes the sender's commitment message. Additionally, we use an instance-dependent resettable witness-indistinguishable argument rWI and resettable-sound witness-indistinguishable arguments of knowledge rsWIAOK (see definitions in Section 3.5.1).

The protocol closely follows the idea of making robust unobfuscatable functions (with a hardcore secret) verifiable, as presented in Section 3.4.1. Non-interactive commitments are replaced with two-message commitments. ZAPs are replaced with instance-dependent rWIs (as defined in Section 3.5.1.1), where the dependance is on the proven statement x . (The latter will be used by the verifier to prove that it acts properly, so they need to be sound only when $x \in \mathcal{L}$, and rWI only when $x \notin \mathcal{L}$). The protocol is described in Figure 3.2, where m is a parameter that controls the number of slots. Similarly to Protocol 3.1, we set $m = 2$ to get constant round resettable-sound

Protocol 3.2

Common Input: $\varphi \in \mathcal{L} \cap \{0, 1\}^n$.

Auxiliary Input to $\mathcal{P}$: $w \in \mathcal{R}_{\mathcal{L}}(\varphi)$.

1. $\mathcal{P}$ samples the first message r for a statistically binding commitment scheme and sends r to $\mathcal{P}$.
2. $\mathcal{V}$ samples a bit $b \leftarrow \{0, 1\}$.
For $i \in \{0, 1\}$, $\mathcal{V}$ samples $g_{k_i} \in \mathcal{G}$ and $vk_i = (h_i, y_i)$, where $h_i \leftarrow \mathcal{HC}_n$ is a randomly chosen hardcore function, and $y_i = \varphi(h_i(k_i))$.
 $\mathcal{V}$ computes $C = \text{Com}_r(b)$, $C_i = \text{Com}_r(k_i)$ and sends $(vk_0, vk_1, C_0, C_1, C)$ to $\mathcal{P}$.
3. Repeat the following function evaluation slot m times:
 - (a) $\mathcal{P}$ samples $q \leftarrow \mathcal{D}(1^n)$ and sends q to $\mathcal{V}$.
 - (b) $\mathcal{V}$ evaluates the functions $a_0 = g_{k_0}(q)$, $a_1 = g_{k_1}(q)$ and sends a_0, a_1 to $\mathcal{P}$.
 - (c) $\mathcal{V}$ proves to $\mathcal{P}$ using an x -dependent rWI^x argument that the following statement:

$$\psi(C_0, C_1, C, q, a_0, a_1) := \left\{ \{C = \text{Com}_r(0)\} \vee \left\{ \begin{array}{l} C_0 = \text{Com}_r(k_0) \\ a_0 = g_{k_0}(q) \end{array} \right\} \right\} \bigwedge \left\{ \{C = \text{Com}_r(1)\} \vee \left\{ \begin{array}{l} C_1 = \text{Com}_r(k_1) \\ a_1 = g_{k_1}(q) \end{array} \right\} \right\}$$

4. $\mathcal{P}$ proves the following statement to $\mathcal{V}$ using an rsWIAOK :
“ $x \in \mathcal{L}$ or $\{\exists z : y_0 = \varphi(z) \vee y_1 = \varphi(z)\}$ ”.

Figure 3.2: A resettably-sound (concurrent) ZK protocol from any one-way function

ZK and $m = n^{\Omega(1)}$ to get a resettably-sound concurrent ZK.

Theorem 48 (resettably-sound (concurrent) ZK from one-way functions). *Protocol 3.2 with $m = 2$ is a constant round resettably-sound ZK protocol, and with $m = n^{\Omega(1)}$, it is a resettably-sound concurrent ZK protocol.*

Corollary 49 (of Theorem 48 and [15]). *There exists a $\tilde{O}(\log n)$ -round resettable ZK argument of knowledge based on one-way functions.*

Corollary 50 (of Theorem 48, [69], and [61]). *There exists a simultaneously-resettable ZK argument of knowledge based on one-way functions.*

The proof of Theorem 48 closely follows the proofs of Lemmas 46, 42, and 43, and is omitted.⁴ Corollary 49 follows by plugging in our one-way function based resettably-sound ZK protocol into the transformation of [15], and Corollary 49 follows by plugging

⁴The only essential different feature is that, to prevent undetectable errors, we use an instance-dependent WI argument with computational rather than statistical soundness, which will induce an additional negligible simulation error $\nu(n)$, where ν is determined by the poly-size verifier $\mathcal{V}^*$ at hand.

in our protocol simultaneously-resettable WI from one way functions [61] into the transformation of [69]. (More accurately, the transformation of [69] is described for ZAPs, but as observed in [61] it actually holds for any simultaneously-resettable WI.) .

3.5.6 A 3-Message Simultaneously-Resettable WI Argument of Knowledge

In this section, we construct a three-message simultaneously-resettable WI proof-of-knowledge protocol based on robust unobfuscatable functions, where knowledge extraction is performed by a non-black-box extractor. As in Section 3.4.1, our protocol will use the idea of turning a single witness statement into a two independent-witnesses statement as done in [58, 78].

Lemma 51. *Protocol 3.3 is a resettable WI argument of knowledge.*

Since the protocol is a three-message protocol, we can apply the [15] transformation, where $\mathcal{V}$ derives its randomness by applying a PRF to the transcript; as a corollary, we get the following theorem:

Corollary 52. *Assuming trapdoor permutations, there exists a three-message simultaneously-resettable WI argument of knowledge (with non-black-box knowledge extraction).*

We now give a proof sketch of the lemma.

Proof of Lemma 51. We start by showing that the protocol is resettable WI. Let

$$(\mathcal{X}, \mathcal{W}_0, \mathcal{W}_1) = \{(x, w_0, w_1) : (x, w_0), (x, w_1) \in \mathcal{R}_{\mathcal{L}}\}$$

be any infinite sequence of instances in $\mathcal{L}$ and corresponding witness pairs. We next consider a sequence of hybrids starting with an hybrid describing an interaction with a prover that uses $w_0 \in \mathcal{W}_0$, and ending with an hybrid describing an interaction with a prover that uses $w_1 \in \mathcal{W}_1$, where both w_0, w_1 , are witnesses for some $x \in \mathcal{X}$. We shall prove that no efficient verifier can distinguish between any two hybrids in the sequence. The list of hybrids is given in Table 4.1. We think of the hybrids as two symmetric sequences: one 0.1-6, starts from witness w_0 , and the other 1.1-6 starts at witness w_1 . We will show that within these sequences the hybrids are indistinguishable, and then we will show that 0.6 is indistinguishable from 1.6.

Protocol 3.3

Common Input: $\varphi \in \mathcal{L} \cap \{0, 1\}^n$.

Auxiliary Input to $\mathcal{P}$: $w \in \mathcal{R}_{\mathcal{L}}(\varphi)$.

1. $\mathcal{P}$ performs the following:

- samples two keys $k_0, k_1 \leftarrow \{0, 1\}^n$ for $f_{k_0}, f_{k_1} \in \mathcal{F}$, and a bit $b \leftarrow \{0, 1\}$.
- samples two hardcore functions $h_0, h_1 \leftarrow \mathcal{HC}_n$ and computes $r_0 = h_0(k_0), r_1 = h_1(k_1)$.
- sends $\mathcal{V}$: $C_0 = \text{Com}(k_0), C_1 = \text{Com}(k_1), C = \text{Com}(b)$, and $h_0, e_0 = w \oplus r_0, h_1, e_1 = w \oplus r_1$.

2. $\mathcal{V}$ performs the following:

- samples randomness $r \leftarrow \{0, 1\}^{\text{poly}(n)}$ for an rZAP, and an input $q \leftarrow \mathcal{D}(1^n)$.
- sends (q, r) to $\mathcal{P}$.

3. $\mathcal{P}$ performs the following:

- computes $a_0 = f_{k_0}(q), a_1 = f_{k_1}(q)$.
- computes an rZAP proof π for the statement:

$$\left\{ \left\{ C = \text{Com}(0) \right\} \vee \left\{ \begin{array}{l} C_0 = \text{Com}(k_0) \\ a_0 = f_{k_0}(q) \\ w \in \mathcal{R}_{\mathcal{L}}(x) \\ e_0 = w \oplus h_0(k_0) \end{array} \right\} \right\} \bigwedge$$

$$i \left\{ \left\{ C = \text{Com}(1) \right\} \vee \left\{ \begin{array}{l} C_1 = \text{Com}(k_1) \\ a_1 = f_{k_1}(q) \\ w \in \mathcal{R}_{\mathcal{L}}(x) \\ e_1 = w \oplus h_1(k_1) \end{array} \right\} \right\}$$

- sends $\mathcal{V}$: a_0, a_1, π .

4. $\mathcal{V}$ verifies the rZAP proof π , and decided whether to accept accordingly.

Figure 3.3: An rWI three-message Argument of Knowledge (implying srWIAOK)

Hybrid 0.1: This hybrid describes a true interaction of a resetting verifier $\mathcal{V}^*$ with an honest prover $\mathcal{P}$ that uses w_0 as a witness for the statement $x \in \mathcal{L}$. In particular, the rZAP uses the witness $((k_0, w_0), (k_1, w_0))$; formally, the witness also includes the randomness for the commitments C_0 and C_1 , but for notational brevity, we shall omit it. In Table 4.1, the witness used in part 0 of the rZAP is referred to as zapw_0 , and the one corresponding to 1 in zapw_1 .

Hybrid 0.2: This hybrid differs from the previous one only in the witness used in the rZAP. Specifically, for the bit b given by C , the witness for the rZAP is set to be $(b, (k_{1-b}, w_0))$, instead of $((k_b, w_0), (k_{1-b}, w_0))$. (Again the witness should include the

hyb	zapw_b	C_b	r_b	$\mathbf{e}_b \oplus r_b$	zapw_{1-b}	C_{1-b}	r_{1-b}	$\mathbf{e}_{1-b} \oplus r_{1-b}$
0.1	(k_b, w_0)	k_b	$h_b(k_b)$	w_0	(k_{1-b}, w_0)	k_{1-b}	$h_{1-b}(k_{1-b})$	w_0
0.2	b	k_b	$h_b(k_b)$	w_0	(k_{1-b}, w_0)	k_{1-b}	$h_{1-b}(k_{1-b})$	w_0
0.3	b	$0^{ k_b }$	$h_b(k_b)$	w_0	(k_{1-b}, w_0)	k_{1-b}	$h_{1-b}(k_{1-b})$	w_0
0.4	b	$0^{ k_b }$	u	w_0	(k_{1-b}, w_0)	k_{1-b}	$h_{1-b}(k_{1-b})$	w_0
0.5	b	$0^{ k_b }$	u	w_1	(k_{1-b}, w_0)	k_{1-b}	$h_{1-b}(k_{1-b})$	w_0
0.6	(k_b, w_1)	k_b	$h_b(k_b)$	w_1	(k_{1-b}, w_0)	k_{1-b}	$h_{1-b}(k_{1-b})$	w_0
1.6	(k_b, w_0)	k_b	$h_b(k_b)$	w_0	(k_{1-b}, w_1)	k_{1-b}	$h_{1-b}(k_{1-b})$	w_1
1.2-5	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$
1.1	(k_b, w_1)	k_b	$h_b(k_b)$	w_1	(k_{1-b}, w_1)	k_{1-b}	$h_{1-b}(k_{1-b})$	w_1

Table 3.1: The sequence of hybrids; the bit b corresponds to the bit commitment C ; the gray cells indicate the difference from the previous hybrid.

randomness for the commitment C , and C_{1-b} , but is omitted from our notation.) Since the rZAP is resettable WI, this hybrid is computationally indistinguishable from the previous one.

Hybrid 0.3: In this hybrid, the commitment C_b is for the plaintext $0^{|k_b|}$, instead of the plaintext k_b . This hybrid is computationally indistinguishable from the previous one due to the computational hiding of the commitment scheme C .

Hybrid 0.4: In this hybrid, instead of sending the verifier the hardcore secret $h_b(k_b)$, it is given a random independent string $u \leftarrow \{0, 1\}^{|h_b(k_b)|}$. Computational indistinguishability of this hybrid and the previous one, follows by the black-box indistinguishability of $\mathcal{F}$ (Definition 21). Indeed, note that any distinguisher here can be turned into a distinguisher against $\mathcal{F}$ and its corresponding hardcore family $\mathcal{HC}$, by treating the oracle as k_b and simulating all the other elements in the experiment.

Hybrid 0.5: In this hybrid, the padded value $\mathbf{e}_b$ is taken to be $w_1 \oplus r_b$, instead of $w_0 \oplus r_b$. Since r_b is now uniform and independent of all other elements, this hybrid induces the exact same distribution as the previous hybrid.

Hybrid 0.6: This hybrid now backtracks, returning to the same experiment as in hybrid 0.1 with the exception that the rZAP witness is now $((k_b, w_1), (k_{1-b}, w_0))$ instead of $((k_b, w_0), (k_{1-b}, w_0))$. This indistinguishability follows exactly as when moving from 0.1 to 0.5 (only backwards).

Hybrids 1.1 to 1.6: These hybrids are symmetric to the above hybrids, only that they start from w_1 instead of w_0 . This means that they end in 1.6 which uses an rZAP witness $((k_b, w_0), (k_{1-b}, w_1))$, which is the same as 0.6, only in reverse order.

Hybrids 0.6 and 1.6 are computationally indistinguishable. This follows directly from the computational hiding of $C = \text{Com}(b)$. Indeed, assume towards contradiction that $\mathcal{V}$ distinguishes the two hybrids. Concretely, denote the probability it outputs 1 on 0.6 by p_0 , and the probability it outputs 1 on 1.6 by p_1 , and assume WLOG that $p_0 - p_1 \geq \varepsilon$, for some noticeable $\varepsilon = \varepsilon(n)$. We can construct a predictor that given a commitment $C = \text{Com}(b)$ to a random bit $b \leftarrow \{0, 1\}$, guesses b with probability $\frac{1+\varepsilon}{2}$. The predictor, samples a random $b' \leftarrow \{0, 1\}$ as a candidate guess for b , and performs the experiment corresponding to 0.6, only that it locates w_0 and w_1 according to b' , rather than the unknown b . If the distinguisher outputs 1, the predictor guesses $b = b'$ and otherwise it guesses $b = 1 - b'$.

Conditioned on $b = b'$, $\mathcal{V}$ is experiencing 0.6, and thus the guess will be correct with probability p_0 ; conditioned on $b = 1 - b'$, $\mathcal{V}$ is experiencing 1.6, and the guess will be right with probability $1 - p_1$. So overall the guessing probability is $\frac{p_0}{2} + \frac{1-p_1}{2} \geq \frac{1}{2} + \frac{\varepsilon}{2}$. This completes the proof that the protocol is resettable WI.

It is left to show that the protocol is an argument of knowledge. Indeed, let $\mathcal{P}^*$ be any prover that convinces the honest verifier of accepting with noticeable probability $\varepsilon = \varepsilon(n)$, then with probability at least $\varepsilon/2$ over its first message, it holds with probability at least $\varepsilon/2$ over the rest of the protocol that $\mathcal{P}^*$ convinces $\mathcal{V}$. Let us call such a prefix **good**. Now for any **good** prefix, we can consider the perfectly binding induced commitment to the bit b , and from the soundness of the rZAP, we get a circuit that with probability at least $\varepsilon/2$ computes correctly the function $f_{k_{1-b}}$, and gives a valid witness $w \in \mathcal{R}_{\mathcal{L}}$, padded with $h_{1-b}(k_{1-b})$. This in particular, means that we can first sample a prefix (hope it is **good**), and then invoke the non-black-box learnability guarantee (Definition 21) to learn $h_{1-b}(k_{1-b})$, and thus also the witness w . (Since we do not know what is the bit b , we will need to construct two circuits for both options of b , and try to extract from both, just as was done in the proof of Claim 3.4.2.) This completes the proof of Lemma 51. $\square$

3.6 From Resettable-Sound ZK back to Unobfuscatable Functions

Our constructions of resettable protocols are based on robust unobfuscatable functions. In the converse direction, we show how to transform any resettable-sound ZK argument

(for NP) to a family of unobfuscatable functions that is not necessarily robust; namely, where extraction is guaranteed only for circuits that compute the function exactly (as defined in [16]).

The high-level idea is to consider the family of functions defined by the resettably-sound ZK verifier's next message function where the key of the function contains the verifier's input: a hard-to-prove statement and random coins. Using resettable soundness we show that a black-box learner cannot produce an accepting protocol transcript. However, given a circuit that computes the function exactly, we show that the ZK simulator can be used to produce an accepting protocol transcript.

Construction 3.6.1. *Let $\text{PRG} : \{0,1\}^n \rightarrow \{0,1\}^{2n}$ be a length-doubling pseudo-random generator and let $\mathcal{L}$ be the image set of PRG. Let $(\mathcal{P}, \mathcal{V})$ be a d -round resettably-sound argument for $\mathcal{L}$ where $\mathcal{V}$ uses m bits of randomness. We construct a family of functions $\{f_k\}_{k \in \{0,1\}^{n+m+1}}$ that implements the honest verifier strategy on some “hard to prove statement”. More precisely, we interpret the key of the function as a triplet $k = (x, r, b)$, where x is an input to PRG, r is randomness for $\mathcal{V}$ and b is a bit. The function f_k is defined as follows:*

1. *Given a special input “STATEMENT”, f_k outputs $y = \text{PRG}(x)$.*
2. *Given a partial protocol transcript consisting of prover messages $\mathbf{ts} = (p_1, \dots, p_i)$ for $i < d$, f_k executes the honest verifier $\mathcal{V}$ with the random tape r and the statement “ $y \in \mathcal{L}$ ” and feeds it with the messages $(p_1, \dots, p_i)$ as the first i prover messages. f_k outputs v_i , the next verifier message generated by $\mathcal{V}$.*
3. *Given a full transcript $\mathbf{ts} = (p_1, \dots, p_d)$, f_k executes $\mathcal{V}_r$ as above. If $\mathcal{V}_r$ accepts, f_k outputs b ; otherwise, it outputs $\perp$.*

We sketch why $\{f_k\}_{k \in \{0,1\}^{n+m+1}}$ is a family of unobfuscatable functions; indeed:

1. **b is black-box unlearnable:** no poly-size algorithm L that is given black-box access to f_k where k is sampled uniformly, can guess b with more than negligible advantage (over a random guess). Indeed, the bit b remains information theoretically hidden, unless L produces a convincing proof for the fact that $y \in \mathcal{L}$. Since L only interacts with f_k as a black-box, we can easily transform L into a resetting adversary $\mathcal{P}^*$ for the resettably-sound ZK argument, which convinces the resettable verifier $\mathcal{V}$ with roughly the same advantage that L has in learning

- b . By the pseudo-randomness guarantee of PRG, $\mathcal{P}^*$ does the same given a statement “ $y \in \mathcal{L}$ ” for a random $y \in \{0,1\}^{2n}$. Since such a statement is false with overwhelming probability, this contradicts the resettable-soundness of $(\mathcal{P}, \mathcal{V})$.
2. **b is non-Black-box learnable:** there exists a PPT algorithm $\mathcal{E}$ that, given any circuit C that computes the function f_k , can learn the value of b . $\mathcal{E}$ uses the code of C to construct an interactive verifier $\mathcal{V}'$ for the resettable-sound ZK protocol. $\mathcal{E}$ then runs the ZK simulator $\mathcal{S}$ with the code of $\mathcal{V}'$ and obtains a transcript $\mathbf{ts}$. It follows from the correctness of the simulation that $\mathbf{ts}$ is, indeed, an accepting transcript. $\mathcal{E}$ runs C on the sequence of prover messages in $\mathbf{ts}$ and obtains the bit b .

Chapter 4

Extractable One-Way Functions

This chapter presents our results regarding the existence of extractable one-way functions. The chapter is based on [32].

4.1 Overview

In this section, we give a high-level overview of the results in this chapter, and of the corresponding techniques. Before that, we briefly recall the basic relevant concepts and main motivation, which were previously described in the introduction.

4.1.1 Basic Concepts and Motivation

A family of functions $\{f_e\}$ is extractable if, any (possibly adversarial) program $\mathcal{A}$ that given e outputs v in the image of f_e has an “extractor” $\mathcal{E}$ that given e and the code of $\mathcal{A}$, outputs a preimage of v . For the notion to be non-trivial, some additional hardness property should be associated with the family. In this chapter, we will consider extractable one-way functions (EOWFs). (In the next chapter we shall also consider extractable collision-resistant hash functions.)

Extractable functions, beyond knowledge assumptions. As discussed in the introduction, extractable functions have been used to circumvent black-box lower-bounds in several applications (of non-interactive nature). In particular, the extractor for any such extractable function must make a non-black-box use of the adversary’s program $\mathcal{A}$. Previously, no construction with an explicit extraction procedure was known. Instead, the existence of such a non-black-box extractor was merely *assumed*.

In light of this, the main question addressed in this chapter is:

Q : *Can we construct extractable functions from standard hardness assumptions?*

Alternatively, can we show that extractable functions cannot exist?

On the role of auxiliary input. We show that the answer to the above question crucially depends on how we model the “auxiliary information” available to the evaluator $\mathcal{A}$ and the extractor $\mathcal{E}$. Let us elaborate. One straightforward formulation of extractable functions requires that, for any possible adversary (modeled as a uniform algorithm) there exists an extractor (again, modeled as a uniform algorithm) that successfully extracts as described above given the adversary’s coin tosses. An alternative is to model both the adversary and the extractor as non-uniform families of deterministic polysize circuits.

However, it turns out that in some applications neither formulation suffices. Indeed, when using extractable functions with other components in a larger cryptographic scheme or protocol, an adversary $\mathcal{A}$ may gather information z from other components and use it as *additional* auxiliary input when evaluating the extractable function. To be useful in these cases, the extractor needs to be able to deal with auxiliary information that’s determined *after* the extractor has been fixed. That is, we require that for any adversary $\mathcal{A}$ there exists an extractor $\mathcal{E}$ such that for any polysize z , and for a randomly chosen key e , whenever $\mathcal{A}(z, e)$ outputs an image v , $\mathcal{E}(z, e)$ output a corresponding preimage of v . In the above, we can either model both the adversary $\mathcal{A}$ and the extractor $\mathcal{E}$ as uniform polytime machines, or as non-uniform machines with polynomial size advice. We call z the *common auxiliary input*, and if $\mathcal{A}$ and $\mathcal{E}$ are non-uniform we refer to their advice as *individual auxiliary input*.

We note that the concept of common auxiliary input appears elsewhere in cryptography. For instance, to make sure that zero-knowledge protocols remain zero-knowledge under sequential composition, the verifier and simulator get common auxiliary input. Indeed, to obtain this standard formulation of zero-knowledge using extractable functions, extractability with common auxiliary input is needed. In other settings, the definition can be relaxed to consider only the case where the common auxiliary input is taken from some specific distribution that captures the “possible” auxiliary information in a given system, see Remark 96 in Chapter 5.

4.1.2 Results

We give two quite different answers to the above question on the existence of extractable one-way functions. On the negative side, following the common belief (first expressed in [113]), we give formal evidence that extractable one-way functions with common auxiliary input of *unbounded* length may not exist:

Theorem 53 (informal). *If there exist indistinguishability obfuscators for a certain class of circuits, then there do not exist extractable one-way functions with respect to common auxiliary-input of unbounded polynomial length.*

This seems to suggest that the concept of auxiliary-input extractable one-way functions (and other concepts that imply it, such as auxiliary-input extractable collision-resistant hashing or SNARKs, discussed in Chapter 5) may be shaky overall, especially in light of the recent candidate indistinguishability obfuscator for all circuits [83].

Still, we show, for the first time, how to construct extractable one-way functions with an explicit extraction procedure with respect to auxiliary-input of *bounded* polynomial length (common or individual), and in particular, with respect to *uniform adversaries*. More specifically, we first give a construction of extractable one-way functions based on publicly-verifiable **P**-delegation schemes:

Theorem 54 (informal). *Assuming one-way functions and publicly-verifiable **P**-delegation, there exist EOWFs with respect to (common or individual) auxiliary-input of bounded polynomial length.*

While the existence of publicly-verifiable **P**-delegation schemes is perhaps not considered as a standard assumption, it is a *falsifiable* assumption [134],¹ with candidates such as CS proofs [128] or SNARGs (see Chapter 5). We view this construction mainly as a proof of concept, showing that ruling out such extractable functions may be a hard task.

Aiming towards a construction from standard assumptions, we formulate a generalized variant of extractable one-way functions (GEOWFs), capturing the properties which make EOWFs useful, and indeed construct bounded-auxiliary-input GEOWFs from relatively standard assumptions. Specifically, our construction relies on the existence of privately-verifiable **P**-delegation, which was recently established by [119], based, for

¹See discussion in [62] on the equivalent concept of 2-message **P**-certificates.

instance, on the Learning with Errors Assumption. We additionally show that the limitation given by Theorem 53 also holds for GEOWFs.

Relying on GEOWFs, we give the first constructions from standard assumptions of 2-message zero-knowledge arguments and 3-message zero-knowledge arguments of knowledge, against verifiers with bounded-auxiliary-input.

Theorem 55 (informal).

1. Assuming (even privately-verifiable) $\mathbf{P}$ -delegation, there exist GEOWFs with respect to (common or individual) auxiliary-input of bounded polynomial length.
2. Assuming GEOWFs, ZAPs [74], and (even 1-hop [89]) homomorphic encryption, there exists a 3-message ZK argument of knowledge against bounded-auxiliary-input verifiers. Assuming the GEOWFs are one-way against subexponential adversaries, there exists a 2-message ZK argument against bounded-auxiliary-input verifiers.

We next elaborate on each of the results, and the techniques used to achieve it.

4.1.3 Impossibility with respect to Unbounded Auxiliary-Input

To introduce the negative result regarding EOWFs with unbounded (common) auxiliary-input, we first recall the notion of *indistinguishability obfuscation* (IO) [16].

Indistinguishability obfuscation. The notion of IO is a relaxation of the virtual-black-box (VBB) notion, for which no impossibility results are known. The notion only requires that it is hard to distinguish an obfuscation of C_0 from an obfuscation of C_1 , for any two circuits C_0 and C_1 of the same size that compute the exact same function.

Garg et al. [83] who proposed a candidate construction of general-purpose obfuscators. They show that, under algebraic assumptions closely related to multilinear maps [66, 81], their construction satisfies the relaxed notion of *indistinguishability obfuscation* (IO) [16], for which no impossibility results are known. Since the emergence of the Garg et al. candidate, IO has been shown to have variety of powerful positive applications, such as functional encryption, public-key encryption from one way functions, deniable encryption, 2-message multi-party computation, and more [42, 82, 83, 117, 147]. Here we show a negative application.

The tension between obfuscation and extractable functions. As discussed in the introduction, the notions of obfuscation and non-black-box extraction stand in clear contrast. In the context of extractable functions with respect to common auxiliary-input, this was already noted by Hada and Tanaka [113], and [94]. Indeed, they note that the common auxiliary-input z may potentially encode an arbitrary circuit to be executed by the adversary in order to produce an image v . The extractor should, thus, be able to efficiently “reverse engineer” such a circuit, in order to figure out a preimage of v . The tension with obfuscation is evident: if z contains obfuscated code that chooses a preimage in some complicated way, it may be impossible to extract from.

The question is how to turn this intuition into a formal impossibility. While VBB obfuscation may be the natural choice, we do not have any evidence that there exist VBB obfuscators for a complicated task such as the one described above (in fact, there is evidence that they do not [31, 103]). We show that general IO suffices to make this intuition rigorous.

Proof idea. We focus on the ‘hardest scenario’, where the auxiliary input z may represent an arbitrary malicious and potentially obfuscated code. Specifically, we consider the following folklore case where z is an obfuscation of a circuit C_k that, given key e for an extractable function f_e , chooses its preimage in an unpredictable way: it applies a pseudo-random function PRF_k to the key, and outputs the result $f_e(\text{PRF}_k(e))$.

An adversary, given such an obfuscated circuit as auxiliary input z , can run it on the key e for the extractable function and always obtain a proper image. The question is whether the extractor, given the same (e, z) , can output a preimage. Intuitively, had we given the extractor black-box access to the circuit C_k , instead of an obfuscation of C_k , it would have to invert the one-way function to obtain such a preimage. Indeed, since the oracle C_k answers any query e' with $f_{e'}(\text{PRF}_k(e'))$, it follows from pseudo-randomness that finding a preimage of $f_e(\text{PRF}_k(e))$ is as hard as finding a preimage of $f_e(u)$, for a uniformly random u .

Can the above intuition be translated to a proof using IO? Indeed, when z is an IO obfuscation $i\mathcal{O}(C_k)$ of the circuit C_k , it is not clear what kind of information leaks on the PRF key k .² Nevertheless, we show that the above intuition can still be fulfilled.

²In fact, formalizing the above intuition is tricky even with VBB, because one has to reduce extraction of one of perhaps many arbitrary pre-images to the task of predicting some deterministic predicate of the PRF key k .

The idea is to consider an alternative to the circuit C_k that computes the same function, but without actually “knowing” the preimage $\text{PRF}_k(e)$. This is achieved using the *puncturing technique* of Sahai and Waters [147].

Specifically, instead of using any PRF family, we use a *puncturable PRF*. In such PRFs it is possible to puncture a given key k at an arbitrary point x^* in the domain of the function. The punctured function $\text{PRF}_{k_{x^*}}$, with punctured key k_{x^*} , preserves functionality at any other point, but hides any information on the point $\text{PRF}_k(x^*)$; namely, the value $\text{PRF}_k(x^*)$ is pseudo-random, even given (x^*, k_{x^*}) . As shown in several recent works [41, 44, 120], such puncturable PRFs follow from the GGM construction [95].

Using a puncturable PRF in the implementation of C_k , we can now show that if the extractor succeeds in finding a preimage of $v = f_e(\text{PRF}_k(e))$, it would also succeed had we provided it with an obfuscation of the alternative circuit $C_{k_e, v}$. The circuit $C_{k_e, v}$ computes the same function as C_k , but in a different way: it only has the punctured key k_e , and has the value $v = f_e(\text{PRF}_k(e))$ directly hardwired into it, so that it does not have to evaluate the PRF in order to compute it. Thus, the fact that the extractor still succeeds follows by the guarantee of indistinguishability obfuscation. However, now by the pseudo-randomness guarantee at the punctured point e , we know that $\text{PRF}_k(e)$ is pseudo random, even given the circuit $C_{k_e, v}$, and thus the extractor can be used to invert the one-way function f_e from scratch.

Finally, we note that since puncturable PRFs can be constructed from one-way functions, and any EOWF is in particular a OWF, it follows that the impossibility of EOWFs is implied by indistinguishability obfuscation without any further assumptions. We also note that the result naturally extends to the notion of generalized EOWFs (presented in more detail in the following subsection).

So, is the knowledge of exponent assumption wrong? In its original formulation [67] and in subsequent formulations [22, 23, 113], the knowledge of exponent assumption (KEA) was not stated with respect to common auxiliary-input, but rather only for individual auxiliary-input (or completely uniform machines), where any $\mathcal{A}$ with advice $z_{\mathcal{A}}$ has an extractor $\mathcal{E}$ with its own advice $z_{\mathcal{E}}$, and the only common extra information is the adversary’s coin tosses and key for the function. In particular, given a non-uniform adversary $\mathcal{A}$ with an obfuscated code as advice $z_{\mathcal{A}}$, the extractor is allowed

to have a different advice $z_{\mathcal{E}}$, representing the “deobfuscated” code. Indeed, our result does not rule out such a notion of extraction (even assuming IO for all circuits).

Our result does not disvalidate the intuition that “the only way” to compute (g^x, h^x) , given (g, h) is by “knowing” x . As we saw, our adversary and auxiliary-input are devised so that x is actually known, but only by an underlying obfuscated computation, and thus cannot be figured out efficiently by an external extractor.

We also note that our result does not rule out extractable functions with respect to common auxiliary input that is taken from specific distributions that may be conjectured to be “benign”, e.g. the uniform distribution (see Remark 96 in Chapter 5).

4.1.4 Constructions with respect to Bounded Auxiliary-Input

We first formulate a generalized version of EOWFs (GEOWFs), and show how GEOWFs can be constructed from standard assumptions. Then, we shall see that, under appropriate conditions, we can leverage the same ideas in order to get standard EOWFs.

Generalized EOWFs. The essence of EOWFs, and what makes them useful, is the asymmetry between a black-box inverter and a non-black-box extractor: an inverter, which only gets a random image $v = f_e(u)$ of an EOWF, cannot find a corresponding preimage u' , whereas a non-black-box extractor, which is given a code that produces such an image, can find a preimage u' . GEOWFs allow to express this asymmetry in a more flexible way. Concretely, a function family $\mathcal{F}$ is now associated with a “hard” binary relation $\mathcal{R}_e^{\mathcal{F}}$ on image-witness pairs $(f_e(u), u')$. Given $v = f_e(u)$ for a random u , it is infeasible to find a witness u' , such that $\mathcal{R}_e^{\mathcal{F}}(v, u') = 1$. In contrast, a non-black-box extractor that is given a code that produces such an image can find such a witness u' .

It is natural to require that the relation $\mathcal{R}_e^{\mathcal{F}}$ is efficiently testable, in this case we say that the GEOWF is *publicly-verifiable*. However, we shall see that GEOWFs are useful, even for hard relations that are not publicly-verifiable. Specifically, we will consider *privately-verifiable* GEOWFs where $\mathcal{R}_e^{\mathcal{F}}(v, u')$ is not efficiently testable given only $(v = f_e(u), u')$, but can be efficiently tested given u in addition.

The main idea. To convey the basic idea behind our constructions of GEOWFs with respect to bounded auxiliary-input, consider the following first attempt. The GEOWF

f is key-less, it is simply a pseudorandom generator stretching inputs of length n to outputs of length $2n$. The relation $\mathcal{R}^{\mathcal{F}}$ contains pairs $(v, \mathcal{M})$ such that the witness $\mathcal{M}$ is a description of a machine of length at most n , and $\mathcal{M}(1^n)$ outputs v . The fact that the relation $\mathcal{R}^{\mathcal{F}}(v, \cdot)$ is hard to satisfy for $v = f(u)$ and a random u , follows from the pseudo-randomness of the output v . Indeed, a truly random output that is indistinguishable from v would have high Kolmogorov complexity. However, given any adversarial program $\mathcal{M}_{\mathcal{A}}$ whose description size is bounded by n and that outputs some $v \in \{0, 1\}^{2n}$, the description of the program $\mathcal{M}_{\mathcal{A}}$ itself is a witness that satisfies the relation $\mathcal{R}^{\mathcal{F}}(v, \mathcal{M}_{\mathcal{A}})$, and thus extraction is trivial.

The main problem is that the time required to test the relation $\mathcal{R}^{\mathcal{F}}$ (even given some preimage of v) is not bounded by any particular polynomial; indeed, the running time of $\mathcal{M}_{\mathcal{A}}$ may be an arbitrary polynomial. One can try to fix this by padding the witness $\mathcal{M}_{\mathcal{A}}$ with 1^t where t is the running time of $\mathcal{M}_{\mathcal{A}}$. However, now the length of the extracted witness depends on the running time of the adversarial program $\mathcal{M}_{\mathcal{A}}$ and is not bounded by any particular polynomial in the length of the image. Such generalized extractable functions do not seem to be as powerful though; in particular, we do not know how to use them for constructing 2-message and 3-message ZK protocols.

A similar problem is encountered in Barak’s zero-knowledge protocol [11], where the entire computation of a malicious verifier is used as the simulation trapdoor. As in the protocol of Barak, Lindell, and Vadhan [18], we get around this problem using a non-interactive proof system that allows for *quick verification* of (possibly long) computations. Instead of computing the output v of the witness program $\mathcal{M}_{\mathcal{A}}$, $\mathcal{R}^{\mathcal{F}}$ will (quickly) verify a proof for the fact that $\mathcal{M}_{\mathcal{A}}(1^n)$ outputs v . That is, $(v, (\mathcal{M}, \pi)) \in \mathcal{R}^{\mathcal{F}}$ only if π is a convincing proof that $\mathcal{M}(1^n) = v$. Intuitively, the soundness of the proof guarantees that the relation is still hard to satisfy. Extraction from a bounded-auxiliary-input adversary $\mathcal{M}_{\mathcal{A}}$ is done by simply computing a proof for its computation.

P-delegation. The proof system required in our constructions is a non-interactive computationally sound proof for deterministic polytime statements, from hereon referred to as a **P-delegation** scheme. More precisely, in a **P-delegation** scheme, the verifier generates, once and for all, an “offline message” σ together with a private verification state τ and sends σ to the prover. Then, the prover can compute a non-interactive proof π for any adaptively chosen statement of the sort: “machine $\mathcal{M}$ out-

puts v within t steps”. We require that the verifier runs in time polynomial in the security parameter n , but only polylogarithmic in t , and the prover runs in time polynomial in (t, n) . We say that a delegation scheme is *publicly-verifiable* if the verification state τ can be published without compromising soundness. Otherwise we say that the scheme is *privately-verifiable*.

As mentioned in Section 4.1.2, while we do have candidates for publicly-verifiable $\mathbf{P}$ -delegation, their security is not based on standard assumptions. In a recent breakthrough result, Kalai, Raz and Rothblum [119] construct a privately verifiable $\mathbf{P}$ -delegation scheme based on any private information retrieval scheme with sub-exponential security. While the scheme of [119] only has non-adaptive soundness, we use standard techniques to get soundness for a statement that is adaptively chosen from a relatively small set of possible statements. This is indeed what is required for our construction (see the body for more details).

GEOWF from $\mathbf{P}$ -delegation. We now sketch how $\mathbf{P}$ -delegation is used in our constructions. We obtain publicly-verifiable (respectively, privately-verifiable) GEOWFs based on publicly-verifiable (respectively, privately-verifiable) delegation. In both cases, the GEOWF f is key-less, it is given as input a seed $\mathbf{s}$ and a random string r . f applies a pseudo-random generator on $\mathbf{s}$ and obtains an image v . f then uses the randomness r to sample an offline message σ together with a verification state τ for a $\mathbf{P}$ -delegation scheme. Finally, f outputs (v, σ) . We assume that if the delegation scheme is publicly-verifiable, the offline message σ includes the verification state τ . Also, if the delegation scheme is privately-verifiable, we assume that τ can be inefficiently computed from σ . (Both assumption are WLOG.)

The relation $\mathcal{R}^{\mathcal{F}}$ contains pairs consisting of an image (v, σ) and witness $(\mathcal{M}, \pi)$, such that the length of $\mathcal{M}$ is much shorter than the length of v and π is an accepting proof for the statement “ $\mathcal{M}(1^n)$ outputs v ”, with respect to the verification state τ corresponding to the offline message σ . Indeed, if the delegation scheme is publicly-verifiable, τ can be efficiently computed from σ , and therefore the relation $\mathcal{R}^{\mathcal{F}}$ is efficiently testable. And if the delegation scheme is privately-verifiable, τ can be efficiently computed given a preimage of (v, σ) that contains the randomness used to sample σ and τ .

Constructing standard EOWFs. We show how to construct a standard (not generalized) EOWF g from a publicly-verifiable GEOWF f . The basic high-level idea is to embed the structure of the GEOWF f and the relation $\mathcal{R}^{\mathcal{F}}$ into the standard EOWF g . For this purpose, g will get as input a string $i \in \{0, 1\}^n$, which intuitively picks one of two branches for computing the function. If $i \neq 0^n$ (which is almost always the case for a random input) the output is computed in the “normal branch”, where g takes an input u for the GEOWF f and outputs $f(u)$. If $i = 0^n$, the output is computed in the “trapdoor branch”, which is almost never taken for a random input, but is used by the extractor. In the trapdoor branch, g takes as input a candidate output v for f and a witness u' for $\mathcal{R}^{\mathcal{F}}(v, \cdot)$. g verifies that $(v, u') \in \mathcal{R}^{\mathcal{F}}$ and if so, it outputs v . Given an adversarial program $\mathcal{M}_{\mathcal{A}}$ that outputs v in the image of f , the extractor for g can invoke the extractor for f , obtain a witness u' such that $(v, u') \in \mathcal{R}^{\mathcal{F}}$, and from this witness construct a valid (trapdoor branch) primage $(i = 0^n, v, u')$ for v .

The above transformation cannot start from a privately-verifiable GEOWF; indeed public-verification is required so to allow the function to efficiently evaluate the relation $\mathcal{R}^{\mathcal{F}}$ in the trapdoor branch. We also note that the above transformation is oversimplified and implicitly assumes that an adversarial evaluator cannot use the trapdoor branch of the function to produce an output that is in the image of g but not in the image of f , in which case extraction may fail. In the body, we show how to avoid this problem by relying on the specific construction of publicly-verifiable GEOWFs from publicly-verifiable **P**-delegation with an extra property (satisfied by existing candidates).

4.1.5 Zero Knowledge against Verifiers with Bounded Auxiliary-Input

We start by describing how to construct 2-message and 3-message zero-knowledge protocols from standard (non-generalized) EOWFs, and then explain how to replace the EOWFs with GEOWFs.

From EOWF to 3-message zero knowledge. The protocol follows the Feige-Lapidot-Shamir *trapdoor paradigm* [76]. Given, say a key-less, EOWF f , the basic idea is to have the verifier send the prover an image $v = f(u)$ of a random element u , which will serve as the trapdoor. The prover would then give a witness-indistinguishable proof-of-knowledge attesting that it either knows a witness w for the proven statement,

or it knows a preimage u' of v . Intuitively, soundness (and actually proof of knowledge) follow from the one-wayness of f and the proof of knowledge property of the WI system. Zero knowledge follows from the extractability of f . Indeed, the simulator, given the code of the verifier, can run the extractor of the EOWF, obtain u , and use it in the WI proof.

Following through on this intuition encounters several difficulties. First, a WI proof of knowledge requires three messages, and thus a first WI prover message must be sent in the first message of the protocol. Furthermore, the WI statement is only determined when the verifier sends v in the second protocol message. Therefore, we must make sure to use a WI proof of knowledge where the first prover message does not depend on the statement. Another basic problem concerns the length of the first WI message. Recall that, in our construction of EOWFs against bounded auxiliary-input adversaries, the function's output is longer than the adversary's advice. Since a cheating verifier may compute v using the first WI message as an advice, we must therefore use a WI system where the length of the first message is independent of the length of the proven statement. We design a WI argument with the required properties based on ZAPs [74] and extractable commitments [143].

An additional potential problem is that a malicious verifier may output an element $\tilde{v}$ outside of the function's image, an event which in general may not be efficiently recognizable, and cause the simulator to fail. This can be solved in a couple of generic ways, below we outline one such solution, based on 1-hop homomorphic encryption. A different approach to the problem, based on ZAPs is described in [28].

From EOWFs to 2-message zero knowledge. In the 2-message protocol, we replace the 3-message WI proof of knowledge with a 2-message WI proof (e.g. a ZAP). However, in the above 3-message protocol, soundness is established by using the proof-of-knowledge property of the WI, whereas 2-message WI proofs of knowledge are not known. Instead, we prove soundness using complexity leveraging. The prover adds to its message a statistically-binding commitment to junk, and proves that either " $x \in \mathcal{L}$ ", or " $f(u) = v$ and the commitment is to u ". We require that the commitment is invertible in some superpolynomial time T , whereas the one-wayness of f still holds against adversaries that run in time $\text{poly}(T)$. Now, an inverter of f can run the cheating prover with a verifier message that contains its input image v , and brute-force break

the commitment to obtain a preimage of v .

Replacing EOWF with GEOWF. We would like to base our zero-knowledge protocols on privately-verifiable GEOWFs (that can be constructed from standard assumptions) instead of on EOWFs. A natural first attempt is to modify the protocol as follows: the verifier sends an image $v = f(u)$, as before, and the prover then gives a WI proof of knowledge attesting that it either knows a witness w for the proven statement, or that it knows, not a preimage, but a witness u' such that $\mathcal{R}^{\mathcal{F}}(v, u') = 1$. The main problem with this first attempt is that the relation $\mathcal{R}^{\mathcal{F}}$ is not publicly-verifiable, and thus the simulator has no way of proving the statement. Another possible problem is that a malicious verifier may output an element outside of the function's image, an event which in general may not be efficiently recognizable. In such a case there is no extraction guarantee, and simulation may fail.

The solution for both problems is to test the relation $\mathcal{R}^{\mathcal{F}}$, and the validity of the verifier's image, using a two-message secure function evaluation protocol, based for example on a 1-hop homomorphic encryption [89]. More concretely, the verifier, in addition to the the function output v , sends an encryption c of the input u . The simulator then homomorphically evaluates a circuit that efficiently computes $\mathcal{R}^{\mathcal{F}}(v, u')$ given u , as well as verifies that indeed $v = f(u)$. The simulator then obtains an evaluated ciphertext $\hat{c}$ that decrypts to 1 (the honest prover will simply simulate an encryption $\hat{c}$ of 1). Finally, the prover (or simulator) sends back $\hat{c}$, and gives a WI proof of knowledge attesting that it either knows a witness w for the proven statement, or that the ciphertext $\hat{c}$ was generated as described. The verifier verifies the WI proof is accepting and that $\hat{c}$ decrypts to 1.

Limitations on two and three message ZK and related work. Three-message zero-knowledge protocols with black-box simulation exist only for trivial languages [98]. The impossibility extends to the case of adversaries with bounded advice of size $n^{\Omega(1)}$, where n is the security parameter (see Section 4.6 for more details). Three-message zero-knowledge protocols were previously based either on the knowledge of exponent assumption [22, 113], on extractable one-way functions [28], or on other extractability assumptions [51]. In all, the simulator uses a non-black extractor that is only assumed to exist, but not explicitly constructed.

Two-message zero-knowledge arguments against adversaries with unbounded poly-

nomial advice exist only for trivial languages (regardless of how simulation is done) [101]. In fact, impossibility extends even to adversaries with bounded advice, provided that the advice string is longer than the verifier’s message. Barak, Lindell, and Vadhan [18] construct a two-message argument that is zero-knowledge as long as the verifier’s advice is shorter than the verifier message by super-logarithmic additive factor. Indeed, our two-message protocol has the same skeleton. However, security of the Barak et al. protocol is only shown assuming existence of $\mathbf{P}$ -delegation schemes (or universal arguments for non-deterministic languages) that are *publicly verifiable*, which as discussed earlier is not considered to be a standard assumption.

4.1.6 More Related Work

Knowledge assumptions in previous work. In the following we briefly survey prior works which relied on different knowledge assumptions.

Damgård [67] first introduced a Knowledge of Exponent Assumption (KEA) to construct a CCA-secure encryption scheme. Later, Hada and Tanaka [113] showed how to use two KEA assumptions to construct the first three-round zero-knowledge argument. Bellare and Palacio [22] then showed that one of the assumptions of [113] was likely to be false, and proposed a modified assumption, using which they constructed a three-round zero-knowledge argument. Bellare and Palacio [23] also note a problem with the use of the basic KEA in Damgård’s CCA application and propose a stronger KEA variant with interactive extraction that allows to overcome the problem.

Abe and Fehr [3] extended the assumption of [22] to construct the first perfect NIZK for $\mathbf{NP}$ with “full” adaptive soundness. Prabhakaran and Xue [145] constructed statistically-hiding sets for trapdoor DDH groups [70] using a new Knowledge of Exponent assumption. Gennaro et al. [87] used another KEA assumption (again with an interactive flavor) to prove that a modified version of the Okamoto-Tanaka key-agreement protocol [136] satisfies perfect forward secrecy against fully active attackers.

More recently, [112] construct constant-round concurrent zero-knowledge from an interactive variant of KEA. In [28], 3-message concurrent zero-knowledge is constructed from a more generic interactive variant of extractable one-way functions.

Knowledge assumptions have also been excessively used to obtain succinct non-interactive arguments (SNARGs) and related primitives. This is discussed in detail in Chapter 5.

Subsequent negative results for extractable functions. The negative result presented in this chapter, in fact, shows that for any candidate EOWF family $\mathcal{F}$, there exists a distribution $\mathcal{Y}_{\mathcal{F}}$, and an adversary $\mathcal{A}$, such that any extractor $\mathcal{E}$ for $\mathcal{A}$, would fail with respect to common auxiliary-input sampled from $\mathcal{Y}_{\mathcal{F}}$. As noted by Boyle and Pass [45], our result can be generalized so that $\mathcal{Y}$ does not depend on $\mathcal{F}$, but only on some upper bound $T_{\mathcal{F}}$ on its running time (by having $\mathcal{Y}$ encode a proper universal circuit). Boyle and Pass further show that, assuming a strengthening of IO called *differing inputs obfuscation* [16], $\mathcal{Y}$ can be made independent of $T_{\mathcal{F}}$ and only depend on its output length $\ell_{\mathcal{F}}$; in particular, elements sampled from $\mathcal{Y}$ can be longer than $\ell_{\mathcal{F}}$. We note that their result does not clash with our positive result for bounded auxiliary-input, in which $\ell_{\mathcal{F}}$ is made longer than the bound on auxiliary inputs. We also note that both ours and Boyle and Pass’ impossibility apply for a specific and rather contrived distribution. No impossibility is yet known for distributions that may be considered “benign”, such as the uniform distribution.

Chapter Organization

In Section 4.2 we give the relevant definitions for EOWF and GEOWF. In Section 4.3, we present the limitation on unbounded auxiliary-input EOWFs based on indistinguishability obfuscation. In Section 4.4, we present the constructions of bounded-auxiliary-input EOWFs and GEOWFs. In Section 4.5.4, we present the zero-knowledge protocols constructed from GEOWFs. In Section 4.6, we discuss relevant black-box lower for EOWFs, and ZK.

4.2 Extractable One-Way Functions

In this section, we define auxiliary-input extractable one-way functions (EOWFs), bounded-auxiliary-input EOWFs, and generalized extractable one-way functions (GEOWFs).

Definition 56 (Auxiliary-input EOWFs [51]). *Let ℓ, ℓ', m be polynomially bounded length functions. An efficiently computable family of functions*

$$\mathcal{F} = \left\{ f_e : \{0, 1\}^{\ell(n)} \rightarrow \{0, 1\}^{\ell'(n)} \mid e \in \{0, 1\}^{m(n)}, n \in \mathbb{N} \right\} ,$$

associated with an efficient (probabilistic) key sampler $\mathcal{K}_{\mathcal{F}}$, is an auxiliary-input EOWF if it is:

1. **One-way:** For any PPT $\mathcal{A}$, polynomial b , security parameter $n \in \mathbb{N}$, and $z \in \{0, 1\}^{b(n)}$:

$$\Pr_{\substack{e \leftarrow \mathcal{K}_{\mathcal{F}}(1^n) \\ u \leftarrow \{0,1\}^{\ell(n)}}} \left[\begin{array}{l} u' \leftarrow \mathcal{A}(e, f_e(u); z) \\ f_e(u') = f_e(u) \end{array} \right] \leq \text{negl}(n) .$$

2. **Extractable:** For any PPT adversary $\mathcal{A}$, there exists a PPT extractor $\mathcal{E}$ such that, for any polynomial b , security parameter $n \in \mathbb{N}$, and $z \in \{0, 1\}^{b(n)}$:

$$\Pr_{e \leftarrow \mathcal{K}_{\mathcal{F}}(1^n)} \left[\begin{array}{cc} v \leftarrow \mathcal{A}(e; z) & u' \leftarrow \mathcal{E}(e; z) \\ \exists u : f_e(u) = v & f_e(u') \neq v \end{array} \right] \leq \text{negl}(n) .$$

Remark 57. Often, an extra image-verification property is required; namely, given v and e , one should be able to efficiently test whether $v \in \text{Image}(f_e)$. Here there are two flavors to consider:

1. *public verifiability*, where to verify an image all that is required is the (public) key e ; and
2. *private verifiability*, where the randomness used to generate e is required to perform image verification.

Our constructions in Section 4.4.2 will not give efficient image verification. When the party generating e is trusted, image-verification can be generically added using NIZKs; alternatively, when applying EOWFs within a protocol, image verification can be enforced using an interactive proof system (this is exemplified in Sections 4.5.4.1, 4.5.4.2).

Bounded auxiliary input. We now define bounded-auxiliary-input EOWFs. Unlike the definition above, where extraction is guaranteed with respect to auxiliary input of any polynomial size b , here b is fixed in advance and the function is designed accordingly. That is, extraction is only guaranteed against adversaries whose advice is bounded by b , whereas their running time may still be an arbitrary polynomial; this, in particular, captures the class of *uniform polytime adversaries*.

For b -bounded auxiliary input, we also define key-less families. While for unbounded auxiliary input, extraction is impossible for key-less families (the adversary may get as auxiliary input a random image, thus forcing the extractor to break one-wayness), for b -bounded auxiliary input, it may be possible, since the output length ℓ' can be larger than the bound b on the auxiliary input. Our constructions will yield such key-less functions.

Definition 58 (b -bounded-auxiliary-input EOWFs). *Let b, ℓ, ℓ', m be polynomially bounded length functions (where ℓ, ℓ', m may depend on b). An efficiently computable family of functions*

$$\mathcal{F} = \left\{ f_e : \{0, 1\}^{\ell(n)} \rightarrow \{0, 1\}^{\ell'(n)} \mid e \in \{0, 1\}^{m(n)}, n \in \mathbb{N} \right\} ,$$

associated with an efficient (probabilistic) key sampler $\mathcal{K}_{\mathcal{F}}$, is a b -bounded auxiliary-input EOWF if it is:

1. **One-way:** *As in Definition 56.*
2. **Extractable against b -bounded adversaries:** *For any PPT adversary $\mathcal{A}$, there exists a PPT extractor $\mathcal{E}$ such that, for any security parameter $n \in \mathbb{N}$, and $z \in \{0, 1\}^{b(n)}$:*

$$\Pr_{e \leftarrow \mathcal{K}_{\mathcal{F}}(1^n)} \left[\begin{array}{cc} v \leftarrow \mathcal{A}(e; z) & u' \leftarrow \mathcal{E}(e; z) \\ \exists u : f_e(u) = v & f_e(u') \neq v \end{array} \right] \leq \text{negl}(n) .$$

*We say that the function is **key-less** if in all the above definitions the key is always set to be the security parameter; namely, $e \equiv 1^n$. In this case, the extraction guarantee always holds (rather than only for a random key).*

Remark 59 (Bounded randomness). *Throughout, we treat any randomness used by the adversary as part of its advice z ; in particular, in the case of bounded advice, we assume that the randomness is bounded accordingly. For many applications, this is sufficient as we can transform any adversary that uses arbitrary polynomial randomness to one that uses bounded randomness, by having it stretch its randomness with a PRG. This approach is applicable, for example, for ZK against b -bounded auxiliary-input verifiers (see Section 4.5), as well as for any application where testing if the adversary breaks the scheme can be done efficiently.*

Remark 60 (Other forms of auxiliary-input).

1. **Individual vs. common auxiliary-input:** *In the above formulation of extractability, the adversary $\mathcal{A}$ (producing an image) and the extractor $\mathcal{E}$ are modeled as uniform PPT machines that obtain the same common auxiliary-input z . This formulation is aligned with the treatment of auxiliary-input in other settings such as zero-knowledge or obfuscation and, as explained in the intro, is instrumental when arguing about extractable functions in the context of a larger system. As also mentioned in the intro, in certain contexts it may be sufficient to consider individual auxiliary-input, where we only require that for any $\mathcal{A}$ with auxiliary-input $z_{\mathcal{A}}$, there exists an extractor $\mathcal{E}$ with auxiliary-input $z_{\mathcal{E}}$. The extractor's $z_{\mathcal{E}}$ may arbitrarily and inefficiently depend on $z_{\mathcal{A}}$, and could be of an arbitrary polynomial size. This weaker notion may be useful in cases where the adversary's auxiliary inputs do not depend on computations that may have taken place in the system before the extractable function is used. Examples include CCA and plaintext-aware encryption with non-uniform security reductions [23, 67]. (We may also consider a definition that allows both individual and common auxiliary-input.)*
2. **Common but “benign” auxiliary-input:** *In the above formulation, it is required that extraction works for a worst-case choice of the common auxiliary-input z . In certain contexts, however, it is sufficient to consider a definition where the common auxiliary input z is drawn from a specific distribution that is conjectured to be ‘benign’, in the sense that it is unlikely to encode a malicious obfuscation. For instance, the distribution can be uniform or an encryption of a random string. Examples where this is sufficient includes essentially all the works on succinct non-interactive arguments (SNARGs), succinct NIZKs, and targeted malleability, which rely on extractable primitives [29, 30, 33, 34, 40, 68, 71, 86, 105, 111, 124, 126, 130] (see also Remark 96 in Chapter 5).*

4.2.1 Generalized Extractable One-Way Functions

The essence of EOWFs, and what makes them useful, is the asymmetry between an inverter and a non-black-box extractor: a black-box inverter that only gets a random image $v = f_e(u)$ cannot find a corresponding preimage u' , whereas a non-black-box

extractor, which is given a code that produces such an image, can find a preimage u' . *Generalized EOWFs* (GEOWFs) allows to express this asymmetry in a more flexible way. Concretely, a function family $\mathcal{F}$ is now associated with a “hard” relation $\mathcal{R}_e^{\mathcal{F}}(f_e(u), u')$ on image-witness pairs $(f_e(u), u') \in \{0, 1\}^{\ell'} \times \{0, 1\}^{\ell+k}$. Given $v = f_e(u)$ for a random u , it is infeasible to find a witness u' , such that $\mathcal{R}_e^{\mathcal{F}}(v, u') = 1$. In contrast, a non-black-box extractor that is given a code that produces such an image can find such a witness u' .

We consider two variants of GEOWFs: The first is *publicly-verifiable GEOWFs*, where for $(v = f_e(u), u')$, the relation $\mathcal{R}_e^{\mathcal{F}}(v, u')$, can be efficiently tested given v and u' only (and the key e if the function is keyed). The second is *privately-verifiable GEOWFs*, where the relation $\mathcal{R}_e^{\mathcal{F}}(v, u')$, might not be efficiently testable given only $(v = f_e(u), u')$, but it is possible to efficiently test the relation given u in addition.

We note that standard EOWFs, as given in Definition 56, fall under the category of publicly-verifiable GEOWFs, where the relation $\mathcal{R}_e^{\mathcal{F}}(v, u)$ simply tests whether $v = f_e(u)$.

Definition 61 (GEOWFs). *An efficiently computable family of functions*

$$\mathcal{F} = \left\{ f_e : \{0, 1\}^{\ell(n)} \rightarrow \{0, 1\}^{\ell'(n)} \mid e \in \{0, 1\}^{m(n)}, n \in \mathbb{N} \right\} ,$$

associated with an efficient (probabilistic) key sampler $\mathcal{K}_{\mathcal{F}}$, is a GEOWF, with respect to a relation $\mathcal{R}_e^{\mathcal{F}}(v, u)$ on triples $(e, v, u') \in \{0, 1\}^{m(n)+\ell'(n)+\ell(n)+k(n)}$, if it is:

1. **$\mathcal{R}^{\mathcal{F}}$ -Hard:** *For any PPT $\mathcal{A}$, polynomial b , security parameter $n \in \mathbb{N}$, and $z \in \{0, 1\}^{b(n)}$:*

$$\Pr_{\substack{e \leftarrow \mathcal{K}_{\mathcal{F}}(1^n) \\ u \leftarrow \{0, 1\}^{\ell(n)}}} \left[\begin{array}{l} u' \leftarrow \mathcal{A}(e, f_e(u); z) \\ \mathcal{R}_e^{\mathcal{F}}(f_e(u), u') = 1 \end{array} \right] \leq \text{negl}(n) .$$

2. **$\mathcal{R}^{\mathcal{F}}$ -Extractable:** *For any PPT adversary $\mathcal{A}$, there exists a PPT extractor $\mathcal{E}$ such that, for any polynomial b , security parameter $n \in \mathbb{N}$, and $z \in \{0, 1\}^{b(n)}$:*

$$\Pr_{e \leftarrow \mathcal{K}_{\mathcal{F}}(1^n)} \left[\begin{array}{l} v \leftarrow \mathcal{A}(e; z) \\ \exists u : f_e(u) = v \end{array} \wedge \begin{array}{l} u' \leftarrow \mathcal{E}(e; z) \\ \mathcal{R}_e^{\mathcal{F}}(f_e(u), u') \neq 1 \end{array} \right] \leq \text{negl}(n) .$$

We further say that the function is

- **Publicly-verifiable** if $\mathcal{R}_e^{\mathcal{F}}(f_e(u), u')$ can be efficiently computed by a tester $\mathcal{T}(e, f_e(u), u')$.
- **Privately-verifiable** if $\mathcal{R}_e^{\mathcal{F}}(f_e(u), u')$ can be efficiently computed by a tester a tester $\mathcal{T}(e, u, u')$.

Bounded auxiliary input GEOWFs (b -bounded auxiliary-input GEOWFs) are defined analogously to b -bounded auxiliary-input-EOWFs. That is, $\mathcal{R}^{\mathcal{F}}$ -hardness is defined exactly as in Definition 61, whereas $\mathcal{R}^{\mathcal{F}}$ -hardness is only against adversaries with auxiliary input of an apriori fixed polynomial size $b(n)$.

Remark 62 (Does $\mathcal{R}^{\mathcal{F}}$ -hardness imply one-wayness). *In principle, $\mathcal{R}^{\mathcal{F}}$ -hardness may not imply one-wayness of $\mathcal{F}$. Although this is not needed for our purposes, we may further require that the relation $\mathcal{R}^{\mathcal{F}}$ includes all pairs $(f_e(u), u)$, and thus ensure that $\mathcal{R}^{\mathcal{F}}$ -hardness does imply one-wayness.*

Remark 63 (GEOWFs vs. Proximity EOWFs). *In [29], a different variant of EOWFs called proximity EOWFs is defined. There a proximity relation $\sim$ is defined on the range of the function. One-wayness is strengthened to require that not only is inverting $f_e(u)$ is hard, but also finding u' such that $f_e(u) \sim f_e(u')$ is hard. Extractability is weakened so that the extractor is allowed to output u' as above, rather than an actual preimage. GEOWF simply allow the relation to be even more general. Any proximity EOWF with relation $\sim$ implies a GEOWF with relation $\mathcal{R}$, such that $\mathcal{R}(f_e(u), u') = 1$ iff $f_e(u) \sim f_e(u')$. In particular, the limitations we show in Section 4.3 on GEOWFs apply to proximity EOWFs as well.*

4.3 From IO to Impossibility of Unbounded-Auxiliary-Input EOWFs

We show that if there exists indistinguishability obfuscation (IO), there do not exist (generalized) auxiliary-input extractable one-way functions.

We start by defining $i\mathcal{O}$ and puncturable PRFs.

4.3.1 Indistinguishability Obfuscation

Indistinguishability obfuscation was introduced in [16] and given a candidate construction in [83], and subsequently in [13, 47].

Definition 64 (Indistinguishability obfuscation [16]). *A PPT algorithm $i\mathcal{O}$ is said to be an indistinguishability obfuscator (INDO) for $\mathcal{C}$, if it satisfies:*

1. **Functionality:** *For any $C \in \mathcal{C}$,*

$$\Pr_{i\mathcal{O}}[\forall x : i\mathcal{O}(C)(x) = C(x)] = 1 \ .$$

2. **Indistinguishability:** *For any class of circuit pairs $\{(C_n^{(1)}, C_n^{(2)}) \in \mathcal{C} \times \mathcal{C}\}_{n \in \mathbb{N}}$, where the two circuits in each pair are of the same size and functionality, it holds that:*

$$\left\{ i\mathcal{O}(C_n^{(1)}) \right\}_{n \in \mathbb{N}} \approx_c \left\{ i\mathcal{O}(C_n^{(2)}) \right\}_{n \in \mathbb{N}} \ .$$

Remark 65 (Efficiently-falsifiable $i\mathcal{O}$). *The assumption that $i\mathcal{O}$ obfuscators exist according to the above (standard) formulation is not efficiently falsifiable in the language of [134]. Specifically, it can be formulated as a cryptographic game [72, 114] between a challenger and an attacker: the attacker submits two circuits $\mathcal{C}_0, \mathcal{C}_1$, gets an obfuscation $i\mathcal{O}(\mathcal{C}_b)$ for a random b , and has to guess b . However, the challenger cannot efficiently check whether the adversarially chosen circuits indeed compute the same function. Thus, it is not efficiently falsifiable.*

We note, however, that for our specific application, we can settle for a restriction of $i\mathcal{O}$ that is falsifiable: we only require that, for some fixed efficiently samplable distribution $\mathcal{D}$ on pairs of circuits with the same functionality, $i\mathcal{O}$ holds for an honestly sampled pair. That is, instead of letting the attacker adversarially choose the circuits, the challenger samples a pair of circuits $\mathcal{C}_0, \mathcal{C}_1$ from $\mathcal{D}$ and hands them to attacker together with $i\mathcal{O}(\mathcal{C}_b)$.

4.3.2 Puncturable PRFs

We next define puncturable PRFs. We consider a simple case of the puncturable PRFs where any PRF might be punctured at a single point. The definition is formulated as in [147].

Definition 66 (Puncturable PRFs). *Let ℓ, m be polynomially bounded length functions. An efficiently computable family of functions*

$$\mathcal{PRF} = \left\{ \text{PRF}_k : \{0, 1\}^{m(n)} \rightarrow \{0, 1\}^{\ell(n)} \mid k \in \{0, 1\}^n, n \in \mathbb{N} \right\},$$

associated with an efficient (probabilistic) key sampler $\mathcal{K}_{\mathcal{PRF}}$, is a puncturable PRF if there exists a puncturing algorithm Punc that takes as input a key $k \in \{0, 1\}^n$, and a point x^* , and outputs a punctured key k_{x^*} , so that the following conditions are satisfied:

1. **Functionality is preserved under puncturing:** *For every $x^* \in \{0, 1\}^{\ell(n)}$,*

$$\Pr_{k \leftarrow \mathcal{K}_{\mathcal{PRF}}(1^n)} [\forall x \neq x^* : \text{PRF}_k(x) = \text{PRF}_{k_{x^*}}(x) \mid k_{x^*} = \text{Punc}(k, x^*)] = 1.$$

2. **Indistinguishability at punctured points:** *The following ensembles are computationally indistinguishable:*

- $\{x^*, k_{x^*}, \text{PRF}_k(x^*) \mid k \leftarrow \mathcal{K}_{\mathcal{PRF}}(1^n), k_{x^*} = \text{Punc}(k, x^*)\}_{x^* \in \{0, 1\}^{m(n)}, n \in \mathbb{N}}$
- $\{x^*, k_{x^*}, u \mid k \leftarrow \mathcal{K}_{\mathcal{PRF}}(1^n), k_{x^*} = \text{Punc}(k, x^*), u \leftarrow \{0, 1\}^{\ell(n)}\}_{x^* \in \{0, 1\}^{m(n)}, n \in \mathbb{N}}$

To be explicit, we include x^* in the distribution; throughout, we shall assume for simplicity that a punctured key k_{x^*} includes x^* in the clear. As shown in [41, 44, 120], the GGM [95] PRF yield puncturable PRFs as defined above.

4.3.3 The Impossibility Result

We now show that if indistinguishability obfuscators exist, there do not exist auxiliary-input EOWFs or generalized EOWFs (GEOWFs) according to Definitions 56, 61:

Theorem 67. *Assuming indistinguishability obfuscation for all circuits, neither EOWFs nor GEOWFs exist, with respect to common auxiliary-input of unbounded polynomial length.*

Before proving the theorem two remarks are in place:

Remark 68 (Implications for other extractable primitives). *GEOWFs are a minimal extractable cryptographic primitive, in the sense that other extractable primitives such*

as extractable collision-resistance hash functions (ECRHs), or succinct non-interactive arguments of knowledge (discussed in Chapter 5) imply them. These implications are invariant with respect to auxiliary-input, and thus our limitation on common auxiliary input also holds with respect to these extractable primitives.

Remark 69 (Auxiliary-input notions that are not ruled out). *The limitation we prove relies critically on the adversary and extractor having common auxiliary-input, and does not if we only require extractability with respect to individual auxiliary-input, as defined in Remark 60. The result does hold if we allow both individual and common auxiliary-input.*

Also, our result does not apply for any distribution on common auxiliary-inputs, but rather shows that some specific auxiliary-input distribution fails extractability. In particular, we do not rule out natural distributions that may be conjectured to be “benign” (see Remark 60), such as the uniform distribution.

To prove the Theorem 67, for any EOWF (respectively, GEOWF) family $\mathcal{F}$, we shall describe an adversary $\mathcal{A}$ and a distribution $\mathcal{Y}$ on auxiliary inputs, such that **any** extractor fails, for auxiliary inputs sampled from $\mathcal{Y}$. For simplicity of exposition, we first concentrate on the case of plain EOWFs, and then show how it directly extends to the case of GEOWFs.

4.3.3.1 The Universal Adversary

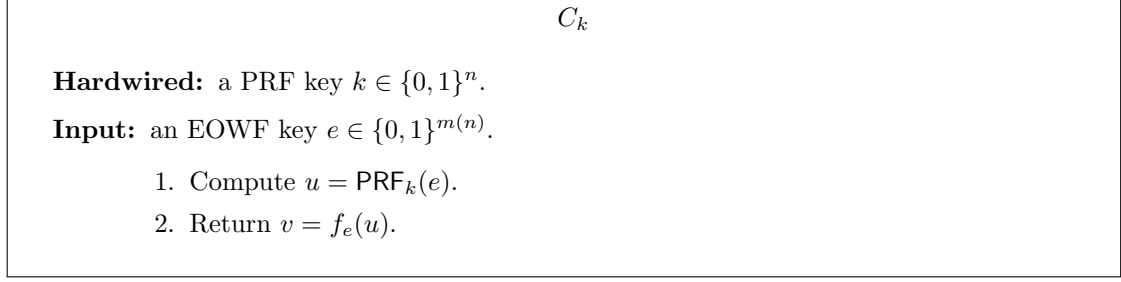
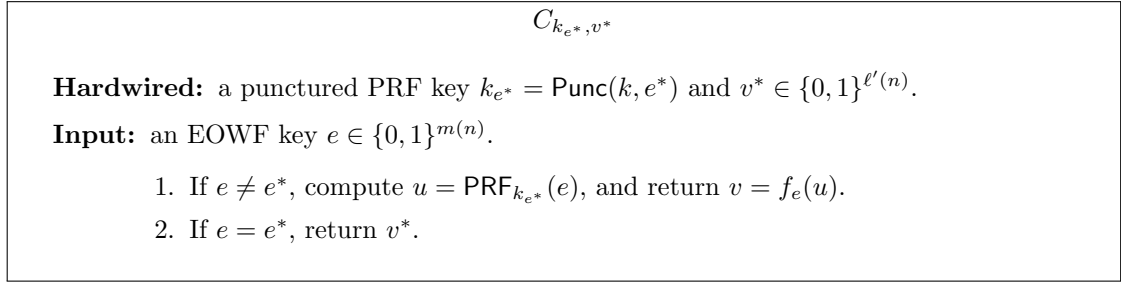
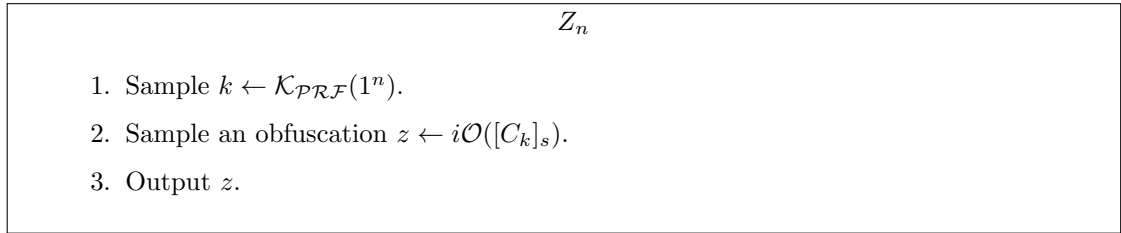
We consider a universal PPT adversary $\mathcal{A}$ that given $(e, z) \in \{0, 1\}^{m(n)} \times \{0, 1\}^{\text{poly}(n)}$, parses z as a circuit and returns $z(e)$.

4.3.3.2 The Auxiliary Input Distribution

Let $\mathcal{F}$ be a family of extractable one-way functions and let $\mathcal{PRF}$ be a puncturable pseudo-random function family. We start by defining two families of circuits

$$\begin{aligned} \mathcal{C} &= \left\{ C_k : \{0, 1\}^{m(n)} \rightarrow \{0, 1\}^{\ell'(n)} \mid k \in \{0, 1\}^n, n \in \mathbb{N} \right\} , \\ \mathcal{C}^* &= \left\{ C_{k_{e^*}, v^*} : \{0, 1\}^{m(n)} \rightarrow \{0, 1\}^{\ell'(n)} \mid k \in \{0, 1\}^n, e \in \{0, 1\}^{m(n)}, v^* \in \{0, 1\}^{\ell'}, n \in \mathbb{N} \right\} . \end{aligned}$$

The circuit C_k , given a key e for an EOWF, applies PRF_k to e , obtains an input u , and returns the result of applying the EOWF f_e to u .

Figure 4.1: The circuit C_k .Figure 4.2: The circuit $C_{k_{e^*}, v^*}$.Figure 4.3: The auxiliary input distribution Z_n .

The circuit $C_{k_{e^*}, v^*}$, has a hardwired PRF key k_{e^*} that was derived from k by puncturing it at the point e^* . In addition, it has hardwired an output v^* to replace the punctured result. In particular, when $v^* = f_{e^*}(\text{PRF}_k(e^*))$ the circuit $C_{k_{e^*}, v^*}$ computes the same function as C_k .

We are now ready to define our auxiliary input distribution $\mathcal{Y} = \{Z_n\}_{n \in \mathbb{N}}$. Let $s = s(n)$ be the maximal size of circuits in either $\mathcal{C}$ or $\mathcal{C}^*$, corresponding to security parameter n , and denote by $[C]_s$ a circuit C padded with zeros to size s . Let $i\mathcal{O}$ be an indistinguishability obfuscator. The distribution Z_n simply consists of an obfuscated (padded) circuit C_k .

4.3.3.3 $\mathcal{A}$ Does Not Have an Extractor

We next show that $\mathcal{A}$ cannot have any extractor $\mathcal{E}$ satisfying Definition 56. In fact, we show a stronger claim; namely, that for the auxiliary input distribution $\mathcal{Y}$, any extractor fails with overwhelming probability.

Proposition 70. *Let $\mathcal{E}$ be any PPT candidate extractor for $\mathcal{A}$ then*

$$\Pr_{\substack{e \leftarrow \mathcal{K}_{\mathcal{F}}(1^n) \\ z \leftarrow Z_n}} \left[\begin{array}{c} v \leftarrow \mathcal{A}(e; z) \\ \exists u : f_e(u) = v \end{array} \wedge \begin{array}{c} u' \leftarrow \mathcal{E}(e; z) \\ f_e(u') \neq v \end{array} \right] \geq 1 - \text{negl}(n) .$$

We note that, since the key e is sampled above independently of the auxiliary input z , the above indeed disproves extractability.

Proof of Proposition 70. First, we note that

$$\Pr_{\substack{e \leftarrow \mathcal{K}_{\mathcal{F}}(1^n) \\ z \leftarrow Z_n}} \left[\begin{array}{c} v \leftarrow \mathcal{A}(e; z) \\ \exists u : f_e(u) = v \end{array} \right] = 1 ;$$

indeed, by the definition of $\mathcal{A}$ and Z_n , and the correctness of $i\mathcal{O}$,

$$\mathcal{A}(e, z) = z(e) = C_k(e) = f_e(\text{PRF}_k(e)) ,$$

where $C_k \in \mathcal{C}$ is the circuit obfuscated in z , i.e. $z = i\mathcal{O}([C_k]_s)$.

Now, assume towards contradiction that, for infinitely many $n \in \mathbb{N}$, the extractor $\mathcal{E}$ successfully outputs a preimage with noticeable probability $\varepsilon(n)$, i.e.

$$\Pr_{\substack{e \leftarrow \mathcal{K}_{\mathcal{F}}(1^n) \\ z \leftarrow Z_n}} \left[\begin{array}{c} u' \leftarrow \mathcal{E}(e; z) \\ f_e(u') = z(e) = f_e(\text{PRF}_k(e)) \end{array} \right] \geq \varepsilon(n) ,$$

where as before, $z = i\mathcal{O}([C_k]_s)$.

Next, for every e^* we consider an alternative distribution $Z_n(e^*, v^*)$ that, instead of sampling a circuit C_k , samples a circuit $C_{k_{e^*, v^*}}$, by first sampling k as usual, and then computing $v^* = f_{e^*}(\text{PRF}_k(e^*))$, and the punctured key k_{e^*} . (Note that $Z_n(e^*, v^*)$ is actually only parameterized by e^* , we add v^* to the notation, to be more explicit.)

We claim that the extractor still succeeds in finding a preimage, i.e.,

$$\Pr_{\substack{e^* \leftarrow \mathcal{K}_{\mathcal{F}}(1^n) \\ z^* \leftarrow Z_n(e^*, v^*)}} \left[\begin{array}{c} u' \leftarrow \mathcal{E}(e^*; z^*) \\ f_{e^*}(u') = z^*(e^*) = v^* = f_{e^*}(\text{PRF}_k(e^*)) \end{array} \right] \geq \varepsilon(n) - \text{negl}(n) .$$

This follows from the fact that, for any e^* and k , C_k and $C_{k_{e^*}, v^*}$ compute the same function, and the $i\mathcal{O}$ indistinguishability guarantee.

Next, we consider another experiment where $Z_n(e^*, v^*)$ is altered to a new distribution $Z_n(e^*, u)$ that, instead of sampling $v^* = f_{e^*}(\text{PRF}_k(e^*))$ in $C_{k_{e^*}, v^*}$, samples $v^* = f_{e^*}(r)$, for an independent random $r \leftarrow \{0, 1\}^\ell$. We claim that

$$\Pr_{\substack{e^* \leftarrow \mathcal{K}_{\mathcal{F}}(1^n) \\ z^* \leftarrow Z_n(e^*, r)}} \left[\begin{array}{c} u' \leftarrow \mathcal{E}(e^*; z^*) \\ f_{e^*}(u') = z^*(e^*) = v^* = f_{e^*}(r) \end{array} \right] \geq \varepsilon(n) - \text{negl}(n) ;$$

indeed, this follows from the fact that $\text{PRF}_k(e^*)$ is pseudo-random, even given the punctured key k_{e^*} .

This means that $\mathcal{E}$ can be used to break the one-wayness of $\mathcal{F}$. Indeed, given a random key e^* , and a challenge $v^* = f_{e^*}(r)$, an inverter can simply sample a punctured k_{e^*} on its own, construct the circuit $C_{k_{e^*}, v^*}$, with its challenge v^* hardwired in, and sample an obfuscation $z^* \leftarrow i\mathcal{O}(C_{k_{e^*}, v^*})$. Finally, it runs $\mathcal{E}(e^*, z^*)$ to invert v^* , with the same probability $\varepsilon(n) - \text{negl}(n)$. $\square$

Extending the result to GEOWFs. The result directly extends to show that no $\mathcal{F}$ can even be a generalized EOWF (GEOWF) with respect to auxiliary input, and any relation $\mathcal{R}^{\mathcal{F}}$. Concretely, we would consider the exact same universal adversary and auxiliary-input distribution $\mathcal{Y}$. The proof goes along the same lines: instead of an extractor that finds a pre-image u of $v = z(e)$, we start from an extractor that finds $u \in \mathcal{R}_e^{\mathcal{F}}(v)$. Then, instead of obtaining an inverter that breaks the one-wayness of $\mathcal{F}$, we obtain an inverter that breaks the $\mathcal{R}^{\mathcal{F}}$ -hardness of $\mathcal{F}$. The proofs follows the exact same arguments. The only thing that should be noted is that when invoking the indistinguishability given by $i\mathcal{O}$, in the first hybrid, and then the indistinguishability given by pseudo-randomness at punctured points, in the second, it can indeed be efficiently tested whether the extractor successfully obtained a witness $u \in \mathcal{R}_e^{\mathcal{F}}(v)$. This is clear in the case that $\mathcal{R}^{\mathcal{F}}$ is publicly-verifiable and also true in the case that it is privately-

verifiable, as in both cases v is computed directly from a pre-image ($\text{PRF}_k(e^*)$, in the first, and r , in the second) that is known to the distinguisher, and which allows testing the relation.

Finally, to deduce Theorem 67, we note that puncturable PRFs can be constructed from one-way functions. Furthermore, EOWF is already a OWF, and any GEOWF with $\mathcal{R}^{\mathcal{F}}$ -hardness implies that $\mathbf{NP} \neq \mathbf{coRP}$, which in conjunction with $i\mathcal{O}$ implies OWFs [132]. Thus, the impossibility of auxiliary-input EOWFs and GEOWFs is implied by indistinguishability obfuscation without any further assumptions.

4.4 Bounded-Auxiliary-Input Extractable One-Way Functions

In this section, we construct bounded-auxiliary-input extractable one-way functions (EOWFs) and bounded auxiliary-input-generalized EOWFs (GEOWFs). Before presenting the construction, we define *non-interactive universal arguments for deterministic computations*, which is the main tool we rely on, and discuss an instantiation based on the delegation scheme of Kalai, Raz, and Rothblum [119].

4.4.1 Non-Interactive Universal Arguments for Deterministic Computations & Delegation

In what follows, we denote by $\mathcal{L}_{\mathcal{U}}$ the universal language [14] consisting of all tuples $(\mathcal{M}, x, t)$ such that $\mathcal{M}$ accepts x within t steps. We denote by $\mathcal{L}_{\mathcal{U}}(T)$ all pairs $(\mathcal{M}, x)$ such that $(\mathcal{M}, x, T) \in \mathcal{L}_{\mathcal{U}}$.

Let $T(n) \in (2^{\omega(\log n)}, 2^{\text{poly}(n)})$ be a computable superpolynomial function. An NIUA system for $\mathbf{Dtime}(T)$ consists of three algorithms $(\mathcal{G}, \mathcal{P}, \mathcal{V})$ that work as follows. The (probabilistic) generator $\mathcal{G}$, given a security parameter 1^n , outputs a *reference string* σ and a corresponding *verification state* τ ; in particular, $\mathcal{G}$ is independent of any statement to be proven later. The honest prover $\mathcal{P}(\mathcal{M}, x; \sigma)$ produces a certificate π for the fact that $(\mathcal{M}, x) \in \mathcal{L}_{\mathcal{U}}(T(n))$. The verifier $\mathcal{V}(\mathcal{M}, x; \pi, \tau)$ verifies the validity of π . Formally, an NIUA system is defined as follows.

Definition 71 (NIUA). *A triple $(\mathcal{G}, \mathcal{P}, \mathcal{V})$ is a non-interactive universal argument system for $\mathbf{Dtime}(T)$ if it satisfies:*

1. **Perfect Completeness:** For any $n \in \mathbb{N}$ and $(\mathcal{M}, x) \in \mathcal{L}_{\mathcal{U}}(T(n))$:

$$\Pr \left[\mathcal{V}(\mathcal{M}, x; \pi, \tau) = 1 \mid \begin{array}{l} (\sigma, \tau) \leftarrow \mathcal{G}(1^n) \\ \pi \leftarrow \mathcal{P}(\mathcal{M}, x; \sigma) \end{array} \right] = 1 .$$

2. **Adaptive soundness for a bounded number of statements:** There is a polynomial b , such that for any polysize prover $\mathcal{P}^*$, $n \in \mathbb{N}$, and set of at most $2^{b(n)}$ false statements $S \subseteq \{0, 1\}^{\text{poly}(n)} \setminus \mathcal{L}_{\mathcal{U}}(T(n))$:

$$\Pr \left[\mathcal{V}(\mathcal{M}, x; \pi, \tau) = 1 \mid \begin{array}{l} (\sigma, \tau) \leftarrow \mathcal{G}(1^n) \\ (\mathcal{M}, x, \pi) \leftarrow \mathcal{P}^*(\sigma) \\ (\mathcal{M}, x) \in S \end{array} \right] \leq \text{negl}(n) .$$

3. **Fast verification and relative prover efficiency:** There exists a polynomial p such that for every $n \in \mathbb{N}$, $t \leq T(n)$, and $(\mathcal{M}, x) \in \mathcal{L}_{\mathcal{U}}(t)$:

- the generator $\mathcal{G}$ runs in time $p(n)$;
- the verifier $\mathcal{V}$ runs in time $p(n, |\mathcal{M}|, |x|)$;
- the prover $\mathcal{P}$ runs in time $p(n, |\mathcal{M}|, |x|, t)$.

The system is said to be **publicly-verifiable** if soundness is maintained when the malicious prover is also given the verification state τ . In this case, we will assume WLOG that the verification state τ appears in the clear in the reference string σ .

Existence and connection to delegation of computation. There are two differences between the notion of delegation for deterministic computations (See, e.g., [119]) and the NIUA notion defined above. The first is that a delegation system is associated with a given language $\mathcal{L}(\mathcal{M})$ for a fixed deterministic machine $\mathcal{M}$, and the corresponding efficiency parameters depend on the worst-case running time $T_{\mathcal{M}}$ of $\mathcal{M}$. In particular, the generator $\mathcal{G}$ depends on $T_{\mathcal{M}}$ as an extra parameter, and the prover's efficiency is polynomial in the worst-case running time $T_{\mathcal{M}}$. The second difference is that only non-adaptive soundness is guaranteed; in particular, the generator's message σ may, in principle, depend on the input x .

Kalai, Raz, and Rothblum [119] show how to construct such a privately verifiable *delegation scheme* for every language in $\mathbf{Dtime}(T) \subseteq \mathbf{EXP}$, assuming subexponentially secure private information retrieval schemes, which can in turn be constructed based on the subexponential Learning with Errors assumption [48].

In order to get a (privately verifiable) NIUA for $\mathbf{Dtime}(T)$, we could potentially use their result with respect to a universal machine and worst-case running time $O(T)$. However, this solution would lack the required prover efficiency, as the prover will always run in time $\text{poly}(T)$, even for machines $\mathcal{M}$ with running time $t_{\mathcal{M}} \ll T$. This is undesired in our case, as we will be interested in T that is super-polynomial. Fortunately, a rather standard transformation does allow to get the required efficiency from their result. Specifically, we could run the generator in their solution to generate a reference string and verification state (σ, τ) for computations of size t for all $t \in \{1, 2, 2^2, \dots, 2^{\log T}\}$, and have the prover and verifier use the right (σ, τ) according to the concrete running time $t_{\mathcal{M}} < T$, guaranteeing that the prover's running time is at most $\text{poly}(2t_{\mathcal{M}})$ as required.

As for adaptivity, in their scheme, the generator does work independently of the input x , but only non-adaptive soundness is shown; namely, soundness is only guaranteed when σ is generated independently of x . To guarantee soundness for adaptively chosen inputs x from a set S of size at most $2^{b(n)}$, we may repeat the above argument $O(b(n))$ times. Assuming that the underlying delegation scheme is secure against provers that run in time $2^{O(b(n))}$ (by choosing the security parameter in the [119] scheme appropriately), the parallel repetition exponentially reduces the soundness error (see e.g., [21]). Then, we can take a union bound over all $2^{b(n)}$ adaptive choices of x and get the required soundness. The $O(b(n))$ -factor hit in succinctness and verification time are still tolerable for our purposes (and still satisfy the above definition).

Theorem 72 (Following from [119]). *Assuming the Learning with Errors Problem is sub-exponentially hard, for any $b(n) = \text{poly}(n)$, and $T(n) \in (2^{\omega(\log n)}, 2^{\text{poly}(n)})$, there exists a (privately-verifiable) NIUA with adaptive soundness for $2^{b(n)}$ statements.*

4.4.2 Constructions

We now present our constructions of bounded-auxiliary-input EOWFs and GEOWFs. We start with the construction of GEOWFs, based on any NIUA. We then give a construction of the standard (rather than generalized) EOWFs based on publicly-verifiable

- Inputs:** (s, r, pad) of respective lengths $(n, n, \ell(n) - 2n)$.
1. Compute $v = \text{PRG}(s)$.
 2. Sample NIUA reference string and verification state $(\sigma, \tau) \leftarrow \mathcal{G}(1^n; r)$.
 3. Output (σ, v) .

Figure 4.4: The function f_n .

NIUAs with an additional key validation property (satisfied by existing candidates).

4.4.2.1 The generalized extractable one-way function

Let $b(n)$ be a polynomial. Let $(\mathcal{G}, \mathcal{P}, \mathcal{V})$ be an NIUA system for $\mathbf{Dtime}(T(n))$ for some function $T(n) \in (2^{\omega(\log n)}, 2^{\text{poly}(n)})$, with adaptive soundness for $2^{b(n)}$ statements. We assume that the system handles statements of the form $(\mathcal{M}, v) \in \{0, 1\}^{b(n)} \times \{0, 1\}^{b(n)+n}$ asserting that “ $\mathcal{M}(1^n)$ outputs v in $T(n)$ steps”. Assume that, $\mathcal{G}(1^n; r)$ uses randomness of size n to output a reference string of polynomial size $m(n)$, and a verification state τ (if the system is publicly-verifiable, then τ appears in σ). Assume that $\mathcal{P}$ outputs certificates π of size $p(n)$. Let PRG be a pseudo random generator stretching n bits to $b(n) + n$ bits. We construct a key-less family of functions $\mathcal{F} = \{f_n\}_{n \in \mathbb{N}}$, consisting of one function $f_n : \{0, 1\}^{\ell(n)} \rightarrow \{0, 1\}^{\ell'(n)}$, for each security parameter n , where $\ell(n) = \max(2n, b(n) + p(n))$ and $\ell'(n) = m(n) + b(n) + n$.

The function is given in Figure 4.4, and is followed by the corresponding relation $\mathcal{R}^{\mathcal{F}}$.

We now define the corresponding relation $\mathcal{R}^{\mathcal{F}} = \{\mathcal{R}_n^{\mathcal{F}}\}_{n \in \mathbb{N}}$ in Figure 4.5, which will be publicly-verifiable (respectively, privately-verifiable) if the NIUA is publicly (respectively, privately verifiable). For simplicity, we assume that the NIUA is such that for every valid reference string σ produced by $\mathcal{G}$, there is a single possible verification state τ (this can always be achieved by adding a commitment to τ inside σ).

Claim 4.4.1. $\mathcal{R}^{\mathcal{F}}$ is publicly-verifiable (respectively privately-verifiable), if $(\mathcal{G}, \mathcal{P}, \mathcal{V})$ is publicly-verifiable (respectively privately-verifiable).

Proof. First, by definition, when $(\mathcal{G}, \mathcal{P}, \mathcal{V})$ is publicly-verifiable, τ can be obtained from σ , NIUA verification can be done efficiently, and thus the relation $\mathcal{R}_n^{\mathcal{F}}$ can be efficiently tested.

Inputs:

$v = f_n(u) = (\sigma, v)$ of respective lengths $(m(n), b(n) + n)$,
 $u' = (\mathcal{M}, \pi, \text{pad})$ of respective lengths $(b(n), p(n), \ell(n) - b(n) - p(n))$.

1. Compute the (unique) verification state τ corresponding to the reference string σ :
2. Run $\mathcal{V}(\mathcal{M}, v, \pi, \tau)$ to verify the statement “ $\mathcal{M}(1^n)$ outputs v in $T(n)$ steps”.
3. Return 1 iff verification passes.

Figure 4.5: The relation $\mathcal{R}_n^{\mathcal{F}}(f_n(u), u')$.

Next, assume that $(\mathcal{G}, \mathcal{P}, \mathcal{V})$ is privately-verifiable. Recall that showing that $\mathcal{R}_n^{\mathcal{F}}$ is privately-verifiable, means that given any preimage u such that $v = f_n(u)$, we can efficiently test $\mathcal{R}_n^{\mathcal{F}}(v, u')$. Indeed, given such a preimage $u = (\mathbf{s}, r, \text{pad})$, we can obtain the generator randomness r , and run $\mathcal{G}(1^n; r)$ to obtain the (unique) verification state τ corresponding to σ , and efficiently test $\mathcal{R}_n^{\mathcal{F}}$. $\square$

Remark 73 (One-wayness vs. $\mathcal{R}^{\mathcal{F}}$ -hardness of $\mathcal{F}$). *The relation $\mathcal{R}^{\mathcal{F}}$ defined above is such that $(f_n(u), u)$ may not satisfy the relation. In particular, this means that $\mathcal{R}^{\mathcal{F}}$ -hardness may not imply one-wayness of $\mathcal{F}$. While this is not needed for our purposes, the relation $\mathcal{R}^{\mathcal{F}}$ can be augmented to also include all pairs $(f_n(u), u)$, and $\mathcal{R}^{\mathcal{F}}$ -hardness will still be preserved; that is, the function we define is one-way in the usual sense.*

We now turn to show that $\mathcal{F}$ is a GEOWF with respect to $\mathcal{R}^{\mathcal{F}}$.

Theorem 74. *The function family $\mathcal{F} = \{f_n\}_{n \in \mathbb{N}}$, given in Figure 4.4 is a GEOWF, with respect to $\mathcal{R}^{\mathcal{F}}$, against $(b(n) - \omega(1))$ -bounded auxiliary-input.*

High-level idea behind the proof. To see that $\mathcal{F}$ is $\mathcal{R}^{\mathcal{F}}$ -hard, note that to break $\mathcal{R}^{\mathcal{F}}$ -hardness, an adversary given a random image (σ, v) , where $v = \text{PRG}(\mathbf{s})$ is of length $b(n) + n$, has to come up with a “small” machine $\mathcal{M}$, whose description length is at most $b(n)$, and a proof that $\mathcal{M}$ outputs v (within a $T(n)$ steps). However, in an indistinguishable world where v is a truly random string, v would almost surely have high Kolomogorov complexity, and a short machine $\mathcal{M}$ that outputs v would not exist. Thus, in this case, the breaker has to produce an accepting proof for a false statement, and violate the soundness of the NIUA.

As for extraction, given a poly-time machine $\mathcal{M}_z$ with short advice z that outputs (σ, v) , where σ is a valid reference string for the NIUA system, the extractor simply com-

puts a proof π for the fact that $\mathcal{M}_z$ outputs v , and outputs the witness $(\mathcal{M}_z, \pi; \text{pad})$. By the completeness of the NIUA system, the proof π is indeed accepting, and the witness satisfies $\mathcal{R}^{\mathcal{F}}$. Furthermore, by the relative prover efficiency of the NIUA, the extractor runs in time that is polynomial in the running time of the adversary $\mathcal{M}_z$.

Proof of Theorem 74. We first show $\mathcal{R}^{\mathcal{F}}$ -hardness, and then show $\mathcal{R}^{\mathcal{F}}$ -extractability.

$\mathcal{R}^{\mathcal{F}}$ -hardness. Assume there exists a breaker $\mathcal{B}$ that, given $v = (\sigma, v)$, where $\sigma \leftarrow \mathcal{G}(1^n)$, and $v \leftarrow \text{PRG}(U_n)$, finds $u = (\mathcal{M}, \pi, \text{pad})$ such that $\mathcal{R}_n^{\mathcal{F}}(v, u) = 1$ with noticeable probability $\varepsilon(n)$. We construct a prover $\mathcal{P}^*$ that breaks the adaptive soundness of the NIUA (for $2^{b(n)}$ statements), with probability $\varepsilon(n) - \text{negl}(n)$. $\mathcal{P}^*$, given σ , first samples on its own $\tilde{v} \leftarrow U_{b(n)+n}$ (independently of σ), and then runs $\mathcal{B}(\sigma, \tilde{v})$ to obtain a machine $\mathcal{M}$ of size $b(n)$, and a proof π .

We first claim that with probability $\varepsilon(n) - \text{negl}(n)$, π is an accepting proof for the statement $(\mathcal{M}, \tilde{v})$ asserting that “ $\mathcal{M}(1^n)$ outputs $\tilde{v}$ in $T(n)$ steps”. Indeed, the view of $\mathcal{B}$ in the above experiment is identical to its real view, except that it gets a truly random $\tilde{v}$, rather than a pseudo-random v that was generated using PRG. Thus, the claim follows by the PRG guarantee.

Next, we note that since $\tilde{v}$ is a $(b(n) + n)$ -long random string, except with negligible probability 2^{-n} , there does not exist $\mathcal{M}$ of size $b(n)$ that outputs $\tilde{v}$. Thus, $\mathcal{P}^*$ produces an accepting proof for one of $2^{b(n)}$ false statements given by the adaptive choice of $\mathcal{M} \in \{0, 1\}^{b(n)}$, and violates the soundness of the NIUA.

$\mathcal{R}^{\mathcal{F}}$ -extractability. We now show $\mathcal{R}^{\mathcal{F}}$ -extractability. We, in fact, show that there is one universal PPT extractor $\mathcal{E}$ that can handle and PPT adversary $\mathcal{M}$ with advice of size at most $b(n) - \omega(1)$. For an adversarial code $\mathcal{M}$ and advice $z \in \{0, 1\}^{b(n) - \omega(1)}$, denote by $\mathcal{M}_z$ the machine that, on input 1^n , runs $\mathcal{M}(1^n; z)$. The extractor $\mathcal{E}$ is given $(\mathcal{M}, z)$, where $\mathcal{M}_z$ has description size at most $b(n)$ and running time at most $t_{\mathcal{M}} < T(n)$, and $\mathcal{M}_z(1^n) = v = (\sigma, v) \in \text{Image}(f_n)$. To compute a witness $u' \in \mathcal{R}^{\mathcal{F}}(v)$, $\mathcal{E}$ computes a certificate π for the fact that “ $\mathcal{M}_z(1^n) = v$ ”, and then outputs $u' = (\mathcal{M}_z, \pi, \text{pad})$. The fact that u' is indeed a valid witness follows directly from the perfect completeness of the scheme. Finally, we note that by the relative prover efficiency of the NIUA the extractor runs in time that is polynomial in the running time $t_{\mathcal{M}}$ of the adversary . $\square$

Remark 75 ($\mathcal{R}^{\mathcal{F}}$ -hardness against superpolynomial adversaries). *In Section 4.5.4.2, we shall require GEOWFs that are $\mathcal{R}^{\mathcal{F}}$ -hard even against adversaries of size $\text{poly}(T(n))$, for some superpolynomial function $T(n)$. Such GEOWFs can be obtained from the above construction, by using a PRG that is secure against $\text{poly}(T(n))$ adversaries, and an NIUA that is sound against such adversaries (such an NIUA can be obtained from [119], based on an appropriately strong private information retrieval scheme).*

4.4.2.2 The standard extractable one-way function

We construct a standard extractable one-way function based on publicly-verifiable NIUAs that have an additional property that says that, in addition to perfect completeness for an honestly chosen reference string σ (which in the publicly-verifiable case is also the verification state), it is also possible to check whether any given σ is valid, or more generally admits perfect completeness. We note that existing candidates for publicly-verifiable NIUAs indeed have this property.³

Definition 76 (NIUA with key validation). *A publicly-verifiable NIUA system is said to have key validation if there exists an efficient algorithm Valid , such that for any $\sigma \in \{0, 1\}^{m(n)}$, if $\text{Valid}(\sigma) = 1$, then the system has perfect completeness with respect to σ . That is, proofs for true statements, generated and verified using σ , are always accepted.*

We now turn to describe the construction, which at a very high-level attempts to embed the structure of the previous GEOWF function and relation into a standard EOWF.

Let $b(n)$ be a polynomial. Let $(\mathcal{G}, \mathcal{P}, \mathcal{V})$ be an NIUA system with the same parameters as in the above GEOWF construction, and with the additional key-validation property. Let PRG be a pseudo random generator stretching n bits to $b(n) + n$ bits.

We construct a key-less family of functions $\mathcal{F} = \{f_n\}_{n \in \mathbb{N}}$, consisting of one function $f_n : \{0, 1\}^{\ell(n)} \rightarrow \{0, 1\}^{\ell'(n)}$, for each security parameter n , where $\ell(n) = 4n + 2b(n) + m(n) + p(n)$ and $\ell'(n) = m(n) + b(n) + n$. The function is given in Figure 4.6.

We now turn to show that $\mathcal{F}$ is an EOWF.

Theorem 77. *The function family $\mathcal{F} = \{f_n\}_{n \in \mathbb{N}}$, given in Figure 4.6 is an EOWF, against $(b(n) - \omega(1))$ -bounded auxiliary-input.*

³Indeed, in Micali's CS proofs, perfect completeness holds with respect to all possible keys for a hash function. In the publicly-verifiable instantiations of the SNARKs constructed in Section 5.4 it is possible to verify the validity of σ using a bilinear map.

Inputs: $(i, (s, r), (\sigma, \mathcal{M}, v, \pi))$ of respective lengths $(n, (n, n), (m(n), b(n), b(n) + n, p(n)))$

- If $i \notin \{0^n, 1^n\}$:
 1. Compute $v^* = \text{PRG}(s)$.
 2. Sample a reference string $\sigma^* \leftarrow \mathcal{G}(1^n; r)$.
 3. Output (v^*, σ^*) .
- If $i = 0^n$:
 1. Perform the following tests:
 - Run $\text{Valid}(\sigma)$ to check the validity of σ ,
 - Run $\mathcal{V}(\mathcal{M}, v, \pi, \sigma)$ to verify the statement “ $\mathcal{M}(1^n)$ outputs v in $T(n)$ steps”,
 If both accept, output (v, σ) .
 2. Otherwise, output $\perp$.
- If $i = 1^n$, output $\perp$.

Figure 4.6: The function f_n .

High-level idea behind the proof. To see that $\mathcal{F}$ is one-way, note that, except with negligible probability, a random image comes from the “normal branch of the function”, where $i \notin \{0^n, 1^n\}$ and includes an honestly sampled σ and a pseudorandom string $v = \text{PRG}(s)$. To invert it, an adversary must either invert $\text{PRG}(s)$, allowing it to produce a “normal branch” preimage, or obtain a short machine $\mathcal{M}$ and an accepting proof π , that $\mathcal{M}$ outputs v , allowing it to produce a “trapdoor branch” preimage. In the first case, the inverter violates the one-wayness of PRG . In the second case, the inverter can be used to break the soundness of the NIUA as in the proof of Theorem 74 (leveraging the fact that a truly random $\tilde{v}$ almost surely cannot be computed by a short machine).

As for extraction, given a poly-time machine $\mathcal{M}_z$ with short advice z that outputs $(\sigma, v) \neq \perp$, by the definition of f_n , σ is a valid reference string for the NIUA system (indeed, $\perp$ is an image that indicates an improper reference string σ , or a non-accepting proof π). In this case, the extractor simply computes a proof π for the fact that $\mathcal{M}_z$ outputs v , and outputs the preimage $(0^n, (0^n, 0^n), (\sigma, \mathcal{M}_z, v, \pi))$. By the completeness of the NIUA system, for a valid σ , the proof π is indeed accepting. By the relative prover efficiency of the NIUA, the extractor runs in time that is polynomial in the running time of the adversary $\mathcal{M}_z$. The only other case to consider is where $\mathcal{M}_z$ outputs $\perp$, in which case producing a preimage is easily done by setting $i = 1^n$.

Proof of Theorem 77. We first show $\mathcal{R}^{\mathcal{F}}$ -hardness, and then show $\mathcal{R}^{\mathcal{F}}$ -extractability.

One-wayness. Assume there exists an inverter $\mathcal{I}$ that, given $v = f_n(u)$, where $u \leftarrow U_{\ell(n)}$, finds a preimage $u' = (i', (s', r'), (\sigma', \mathcal{M}', v', \pi'))$ with noticeable probability $\varepsilon(n)$. We construct a prover $\mathcal{P}^*$ that breaks the adaptive soundness of the NIUA (for $2^{b(n)}$ statements), with probability $\varepsilon(n) - \text{negl}(n)$. $\mathcal{P}^*$ is defined as in the proof of Theorem 74: given σ , it first samples on its own $\tilde{v} \leftarrow U_{b(n)+n}$ (independently of σ), and then runs $\mathcal{I}(\sigma, \tilde{v})$ to obtain $u' = (i', (s', r'), (\sigma', \mathcal{M}', v', \pi'))$.

Claim 4.4.2. *With probability $\varepsilon(n) - \text{negl}(n)$, π' is an accepting proof, with respect to σ , for the statement $(\mathcal{M}', v)$, attesting that “ $\mathcal{M}'(1^n)$ outputs $\tilde{v}$ in $T(n)$ steps”.*

The claim will conclude the proof of one-wayness since, as in the proof of Theorem 74, except with negligible probability, there does not exist a machine $\mathcal{M}'$ of size $b(n)$ that outputs $\tilde{v}$ which is a $(b(n)+n)$ -long random string. This means that $\mathcal{I}$ outputs an accepting proof for one of $2^{b(n)}$ false statements (given different $\mathcal{M}' \in \{0, 1\}^{b(n)}$), and violates the soundness of the NIUA.

Proof. To prove the claim, we first consider an hybrid experiment where $\mathcal{I}$ samples a pseudorandom $v \leftarrow \text{PRG}(U_n)$ instead of a truly random $\tilde{v}$. By the PRG guarantee, we know that the probability of outputting $(\mathcal{M}', \pi)$ as required by the claim changes at most by a negligible amount $\text{negl}(n)$. Next we note that the view of $\mathcal{I}$ in the hybrid experiment is identical to its view in the real world where it receives a random image $v = (\sigma, v)$. Furthermore, whenever $\mathcal{I}$ finds a preimage $u' = (i', (s', r'), (\sigma', \mathcal{M}', v', \pi'))$ of v such that $i' = 0^n$, by the definition of f_n , $(\sigma', v') = (\sigma, v)$, and π' must be an accepting proof for the statement $(\mathcal{M}', v' = v)$, with respect to $\sigma' = \sigma$.

Since we know that $\mathcal{I}$ inverts the function with probability $\varepsilon(n)$, it thus suffices to show that the preimage it finds is such that $i = 0^n$, except with negligible probability. Indeed, whenever $\mathcal{I}$ finds a preimage such that $i' \notin \{0^n, 1^n\}$, by the definition of f_n , it inverts $v = \text{PRG}(s)$, contradicting the one-wayness of PRG. Also, a preimage of (σ, v) cannot have $i' = 1^n$, assuming $(\sigma, v) \neq \perp$, which is the case with overwhelming probability. This concludes the proof of the claim. $\square$

Extractability. We show that there is one universal PPT extractor $\mathcal{E}$ that can handle and PPT adversary $\mathcal{M}$ with advice of size at most $b(n) - \omega(1)$. The proof is similar to the extractability proof of Theorem 74. For an adversarial code $\mathcal{M}$ and advice

$z \in \{0, 1\}^{b(n)-\omega(1)}$, we denote by $\mathcal{M}_z$ the machine that, on input 1^n , runs $\mathcal{M}(1^n; z)$. The extractor $\mathcal{E}$ is given $(\mathcal{M}, z)$, where $\mathcal{M}_z$ has description size at most $b(n)$ and running time at most $t_{\mathcal{M}} < T(n)$, and $\mathcal{M}_z(1^n) = (\sigma, v) \in \text{Image}(f_n)$.

If $(\sigma, v) \neq (0^{m(n)}, 0^{b(n)+n})$, we know that σ must be valid, in which case $\mathcal{E}$ computes a certificate π for the fact that “ $\mathcal{M}_z(1^n) = v$ ”, and then outputs the preimage $u' = (0^n, (0^n, 0^n), (\sigma, \mathcal{M}_z, v, \pi))$. The fact that u' is indeed a valid preimage follows directly from the perfect completeness of the scheme, for a valid σ . If $(\sigma, v) = (0^{m(n)}, 0^{b(n)+n})$, the extractor outputs the preimage $u' = (1^n, (0^n, 0^n), (0^{m(n)}, 0^{b(n)}, 0^{b(n)+n}, 0^{p(n)}))$.

Finally, we note that by the relative prover efficiency of the NIUA the extractor runs in time that is polynomial in the running time $t_{\mathcal{M}}$ of the adversary . $\square$

4.5 2-Message and 3-Message Zero Knowledge against Bounded-Auxiliary-Input Verifiers

In this section, we define and construct two and three message ZK arguments against verifiers with bounded auxiliary input, based on GEOWFs. We start by presenting the definition of such ZK arguments, and two tools which will be of use. Then, we move on to describe our constructions.

4.5.1 Definition

The standard definition of zero knowledge [94, 106] considers adversarial verifiers with non-uniform auxiliary input of arbitrary polynomial size. We consider a relaxed notion of zero knowledge against verifiers that have bounded non-uniform advice, but arbitrary polynomial running time. This relaxed notion, in particular, includes zero knowledge against uniform verifiers (sometimes referred to as *plain zero knowledge* [18]).

Concretely, we shall focus on PPT verifiers $\mathcal{V}^*$ having advice z of size at most $b(n)$, and using an arbitrary polynomial number of random coins.

Definition 78. *An argument system $(\mathcal{P}, \mathcal{V})$ for an NP relation $\mathcal{R}_{\mathcal{L}}(\varphi, w)$ is zero knowledge against verifiers with b -bounded advice if for every PPT verifier $\mathcal{V}^*$, there exists a PPT simulator $\mathcal{S}$ such that:*

$$\{(\mathcal{P}(w), \mathcal{V}^*(z))(\varphi)\}_{\substack{(\varphi, w) \in \mathcal{R}_{\mathcal{L}} \\ z \in \{0, 1\}^{b(|\varphi|)}}} \approx_c \{\mathcal{S}(z, \varphi)\}_{\substack{(\varphi, w) \in \mathcal{R}_{\mathcal{L}} \\ z \in \{0, 1\}^{b(|\varphi|)}}},$$

where computational indistinguishability is with respect to arbitrary non-uniform distinguishers.

Remark 79 (universal simulator). *In the above definition, each PPT $\mathcal{V}^*$ is required to have a designated PPT simulator $\mathcal{S}_{\mathcal{V}^*}$. Our constructions will, in fact, guarantee the existence of one universal simulator $\mathcal{S}$ that, in addition to (z, φ) , is also given the code of $\mathcal{V}^*$ and a bound $1^{t_{\mathcal{V}^*}}$ on the running time of $\mathcal{V}^*(\varphi; z)$, and simulates $\mathcal{V}^*$'s view. Moreover, the running time of $\mathcal{S}$ is bounded by some (universal) polynomial $\text{poly}(t_{\mathcal{V}^*})$ in the running time of $\mathcal{V}^*$. We note that, in ZK with unbounded polynomial auxiliary input, such universality follows automatically by considering the universal machine and auxiliary input $(\mathcal{V}^*, 1^{t_{\mathcal{V}^*}})$. In our context, however, this does not hold since $t_{\mathcal{V}^*}$ is unbounded and can be larger than the bound b on the size of the advice.*

4.5.2 WI Proof of Knowledge

with an Instance-Independent First Message

In this section, we define and construct 3-message WI proofs of knowledge with an instance-independent first message, which will be used in our construction of a 3-message ZK argument of knowledge. In such proof systems, the prover's first message is completely independent of the statement and witness $(\varphi, w) \in \mathcal{R}_{\mathcal{L}}$ to be proven; in particular, it is of fixed polynomial length in a security parameter n , independently of $|\varphi, w|$.

Classical WIPOK protocols do not satisfy this requirement. For example, in the classical Hamiltonicity protocol [38], the first message is independent of the witness w , but does depend on the statement φ . In Lapidot and Shamir's Hamiltonicity variant [123], the first message is independent of (φ, w) themselves, but does depend on $|\varphi, w|$ (see details in [137]). ZAPs do satisfy the independence requirement (as there is no first prover message at all), but they do not constitute a proof of knowledge.

We show that, using ZAPs, and 3-message extractable commitments, we can obtain a WIPOK where the first (prover) message is completely independent of (φ, w) , even of their length, and the second (verifier) message only depends on $|\varphi|$.

Definition 80 (WIPOK with instance-independent first message). *Let $(\mathcal{P}, \mathcal{V})$ be a 3-message proof system for $\mathcal{L}$ with messages (α, β, γ) ; we say it is a WIPOK with instance-independent first message, if it satisfies:*

1. **Completeness with first message independence:** For any $\varphi \in \mathcal{L} \cap \{0, 1\}^\ell$, $w \in \mathcal{R}_{\mathcal{L}}(\varphi)$, $n \in \mathbb{N}$:

$$\Pr \left[\mathcal{V}(\varphi, \alpha, \beta, \gamma; r') = 1 \mid \begin{array}{l} \alpha \leftarrow \mathcal{P}(1^n; r) \\ \beta \leftarrow \mathcal{V}(\ell, \alpha; r') \\ \gamma \leftarrow \mathcal{P}(\varphi, w, \alpha, \beta; r) \end{array} \right] = 1 ,$$

where $r, r' \leftarrow \{0, 1\}^{\text{poly}(n)}$ are the randomness used by $\mathcal{P}$ and $\mathcal{V}$.

The honest prover's first message α is of length n , independently of the length of the statement and witness (φ, w) .

2. **Adaptive witness-indistinguishability:** for any deterministic polysize verifier $\mathcal{V}^*$ and all $n \in \mathbb{N}$:

$$\Pr \left[\mathcal{V}^*(\varphi, \alpha, \beta, \gamma) = b \mid \begin{array}{l} \alpha \leftarrow \mathcal{P}(1^n; r) \\ \varphi, w_0, w_1, \beta \leftarrow \mathcal{V}^*(\alpha) \\ \gamma \leftarrow \mathcal{P}(\varphi, w_b, \alpha, \beta; r) \end{array} \right] \leq \frac{1}{2} + \text{negl}(n) ,$$

where $b \leftarrow \{0, 1\}$, $r \leftarrow \{0, 1\}^{\text{poly}(n)}$ is the randomness used by $\mathcal{P}$, and $w_0, w_1 \in \mathcal{R}_{\mathcal{L}}(\varphi)$.

3. **Adaptive proof of knowledge:** there is a PPT extractor $\mathcal{E}$, such that, for any polynomial $\ell = \ell(n)$, all $n \in \mathbb{N}$, and any deterministic prover $\mathcal{P}^*$:

$$\begin{array}{l} \text{if } \Pr \left[\mathcal{V}(\varphi, \alpha, \beta, \gamma; r') = 1 \mid \begin{array}{l} \alpha \leftarrow \mathcal{P}^* \\ \beta \leftarrow \mathcal{V}(\ell(n), \alpha; r') \\ \varphi, \gamma \leftarrow \mathcal{P}^*(\alpha, \beta) \end{array} \right] \geq \varepsilon , \\ \text{then } \Pr \left[\begin{array}{l} w \leftarrow \mathcal{E}^{\mathcal{P}^*}(1^{1/\varepsilon}, \varphi, \alpha, \beta, \gamma) \\ w \notin \mathcal{R}_{\mathcal{L}}(\varphi) \end{array} \mid \begin{array}{l} \alpha \leftarrow \mathcal{P}^* \\ \beta \leftarrow \mathcal{V}(\ell(n), \alpha; r') \\ \varphi, \gamma \leftarrow \mathcal{P}^*(\alpha, \beta) \\ \mathcal{V}(\varphi, \alpha, \beta, \gamma; r') = 1 \end{array} \right] \leq \text{negl}(n) , \end{array}$$

where $\varphi \in \{0, 1\}^{\ell(n)}$, and $r' \leftarrow \{0, 1\}^{\text{poly}(n)}$ is the randomness used by $\mathcal{V}$.

Construction from ZAPs. We now show how to use ZAPs and extractable commitments to construct a WIPOK with the required properties. As mentioned above, ZAPs already have the required independence, but they do not provide POK. The high-level idea is to add the POK feature to ZAPs, while maintaining the required instance-independence. This can be done by having the prover commit to a random string r using a 3-message extractable commitment (e.g., as formalized in [143]), and then sending, as the third message, the padded witness $w \oplus r$ along with a ZAP proof that it was computed correctly. While the first message is independent of φ, w it does depend on the length $|w|$; this is naturally solved by committing to a seed s of fixed length and later deriving r using a PRG.

Intuitively, extraction of the witness is now possible by extracting r (or s) from the committing prover. To ensure WI we use the idea of turning a single witness statement into a two independent-witnesses statement as done in [58, 78], and in Section 3.5.6.

In what follows, we denote by $(\mathcal{C}, \mathcal{R})$ the committer and receiver algorithms of a perfectly-binding 3-message extractable commitment protocol, and we denote by $\vec{C} = (C^{(1)}, C^{(2)}, C^{(3)})$ its three messages. We further require that extraction is possible given any two valid transcripts $\vec{C}, \vec{C}'$ that share the same first message. Such an extractable commitment can be constructed from any perfectly-binding non-interactive commitment, see e.g. [143].

Lemma 81. *Protocol 4.7 is a 3-message WIPOK with instance-independent first message.*

We next prove the lemma. The proof is an adaptation of the proof of Lemma 51 from Chapter 3.

Proof. We start by showing that the protocol is WI. Let

$$(\bar{\varphi}, \bar{w}_0, \bar{w}_1) = \{(\varphi, w_0, w_1) : (\varphi, w_0), (\varphi, w_1) \in \mathcal{R}_{\mathcal{L}}\}$$

be any infinite sequence of instances in $\mathcal{L}$ and corresponding witness pairs. We next consider a sequence of hybrids starting with an hybrid describing an interaction with a prover that uses $w_0 \in \bar{w}_0$, and ending with an hybrid describing an interaction with a prover that uses $w_1 \in \bar{w}_1$, where both w_0, w_1 , are witnesses for some $\varphi \in \bar{\varphi}$. We shall prove that no efficient verifier can distinguish between any two hybrids in the sequence.

Protocol 4.7

Common Input: security parameter n , and $\varphi \in \mathcal{L} \cap \{0, 1\}^{\ell(n)}$.

Auxiliary Input to $\mathcal{P}$: $w \in \mathcal{R}_{\mathcal{L}}(\varphi)$.

1. $\mathcal{P}$ samples seeds $s_0, s_1 \leftarrow \{0, 1\}^{\sqrt{n}}$, and a bit $b \leftarrow \{0, 1\}$, and sends the first commitment message to each of the three $(C_0^{(1)}, C_1^{(1)}, C^{(1)}) \leftarrow (\mathcal{C}(s_0), \mathcal{C}(s_1), \mathcal{C}(b))$, where $|(C_0^{(1)}, C_1^{(1)}, C^{(1)})| = n$.^a
2. $\mathcal{V}$, given the length of the statement $\ell = |\varphi|$, samples randomness $r \leftarrow \{0, 1\}^{\text{poly}(n)}$ for a ZAP, and receiver messages $(C_0^{(2)}, C_1^{(2)}, C^{(2)}) \leftarrow (\mathcal{R}(C_0^{(1)}), \mathcal{R}(C_1^{(1)}), \mathcal{R}(C^{(1)}))$, and sends $(r, C_0^{(2)}, C_1^{(2)}, C^{(2)})$ to $\mathcal{P}$.
3. $\mathcal{P}$, given (φ, w) , now performs the following:
 - computes the third committer messages $(C_0^{(3)}, C_1^{(3)}, C^{(3)}) \leftarrow (\mathcal{C}(s_0, C_0^{(2)}), \mathcal{C}(s_1, C_1^{(2)}), \mathcal{C}(b, C^{(2)}))$.
 - computes $a_0 = w \oplus \text{PRG}(s_0), a_1 = w \oplus \text{PRG}(s_1)$.
 - computes a ZAP proof π for the statement:

$$\left\{ \left\{ \vec{C} = \mathcal{C}(0, C^{(2)}) \right\} \vee \left\{ \begin{array}{l} \vec{C}_0 = \mathcal{C}(s_0, C_0^{(2)}) \\ a_0 = w \oplus \text{PRG}(s_0) \\ w \in \mathcal{R}_{\mathcal{L}}(\varphi) \end{array} \right\} \right\} \wedge \left\{ \left\{ \vec{C} = \mathcal{C}(1, C^{(2)}) \right\} \vee \left\{ \begin{array}{l} \vec{C}_1 = \mathcal{C}(s_1, C_1^{(2)}) \\ a_1 = w \oplus \text{PRG}(s_1) \\ w \in \mathcal{R}_{\mathcal{L}}(\varphi) \end{array} \right\} \right\}$$

- sends $C_0^{(3)}, C_1^{(3)}, C^{(3)}, a_0, a_1, \pi$.
4. $\mathcal{V}$ verifies the ZAP proof π , the validity of the commitments transcripts, and decides whether to accept accordingly.

^aThe commitment to b does not have to be extractable; however, we use the same commitment scheme to avoid extra notation.

Figure 4.7: A 3-message WIPOK with instance-independent first message

The list of hybrids is given in Table 4.1. We think of the hybrids as two symmetric sequences: one 0.1-6, starts from witness w_0 , and the other 1.1-6 starts at witness w_1 . We will show that within these sequences the hybrids are indistinguishable, and then we will show that 0.6 is indistinguishable from 1.6.

Hybrid 0.1: This hybrid describes a true interaction of a malicious verifier $\mathcal{V}^*$ with an honest prover $\mathcal{P}$ that uses w_0 as a witness for the statement $x \in \mathcal{L}$. In particular, the ZAP uses the witness $((s_0, w_0), (s_1, w_0))$; formally, the witness also includes the randomness for the commitments $\vec{C}_0$ and $\vec{C}_1$, but for notational brevity, we shall omit it. In Table 4.1, the witness used in part 0 of the ZAP is referred to as zapw_0 , and the

hyb	zapw_b	$\vec{C}_b$	r_b	$a_b \oplus r_b$	zapw_{1-b}	$\vec{C}_{1-b}$	r_{1-b}	$a_{1-b} \oplus r_{1-b}$
0.1	(s_b, w_0)	s_b	$\text{PRG}_b(s_b)$	w_0	(s_{1-b}, w_0)	s_{1-b}	$\text{PRG}(s_{1-b})$	w_0
0.2	b	s_b	$\text{PRG}_b(s_b)$	w_0	(s_{1-b}, w_0)	s_{1-b}	$\text{PRG}(s_{1-b})$	w_0
0.3	b	$0^{ s_b }$	$\text{PRG}_b(s_b)$	w_0	(s_{1-b}, w_0)	s_{1-b}	$\text{PRG}(s_{1-b})$	w_0
0.4	b	$0^{ s_b }$	u	w_0	(s_{1-b}, w_0)	s_{1-b}	$\text{PRG}(s_{1-b})$	w_0
0.5	b	$0^{ s_b }$	u	w_1	(s_{1-b}, w_0)	s_{1-b}	$\text{PRG}(s_{1-b})$	w_0
0.6	(s_b, w_1)	s_b	$\text{PRG}_b(s_b)$	w_1	(s_{1-b}, w_0)	s_{1-b}	$\text{PRG}(s_{1-b})$	w_0
1.6	(s_b, w_0)	s_b	$\text{PRG}_b(s_b)$	w_0	(s_{1-b}, w_1)	s_{1-b}	$\text{PRG}(s_{1-b})$	w_1
1.2-5	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$
1.1	(s_b, w_1)	s_b	$\text{PRG}_b(s_b)$	w_1	(s_{1-b}, w_1)	s_{1-b}	$\text{PRG}(s_{1-b})$	w_1

Table 4.1: The sequence of hybrids; the bit b corresponds to the bit commitment $\vec{C}$; the gray cells indicate the difference from the previous hybrid.

one corresponding to 1 in zapw_1 .

Hybrid 0.2: This hybrid differs from the previous one only in the witness used in the ZAP. Specifically, for the bit b given by $\vec{C}$, the witness for the ZAP is set to be $(b, (s_{1-b}, w_0))$, instead of $((s_b, w_0), (s_{1-b}, w_0))$. (Again the witness should include the randomness for the commitment $\vec{C}$, and $\vec{C}_{1-b}$, but is omitted from our notation.) Since the ZAP is WI, this hybrid is computationally indistinguishable from the previous one.

Hybrid 0.3: In this hybrid, the commitment $\vec{C}_b$ is for the plaintext $0^{|s_b|}$, instead of the plaintext s_b . This hybrid is computationally indistinguishable from the previous one due to the computational hiding of the commitment scheme $\vec{C}$.

Hybrid 0.4: In this hybrid, instead of padding with $\text{PRG}(s_b)$, padding is done with a random independent string $u \leftarrow \{0, 1\}^{|\text{PRG}(s_b)|}$. Computational indistinguishability of this hybrid and the previous one, follows pseudorandomness.

Hybrid 0.5: In this hybrid, the padded value a_b is taken to be $w_1 \oplus r_b$, instead of $w_0 \oplus r_b$. Since r_b is now uniform and independent of all other elements, this hybrid induces the exact same distribution as the previous hybrid.

Hybrid 0.6: This hybrid now backtracks, returning to the same experiment as in hybrid 0.1 with the exception that the ZAP witness is now $((s_b, w_1), (s_{1-b}, w_0))$ instead of $((s_b, w_0), (s_{1-b}, w_0))$. This indistinguishability follows exactly as when moving from 0.1 to 0.5 (only backwards).

Hybrids 1.1 to 1.6: These hybrids are symmetric to the above hybrids, only that they start from w_1 instead of w_0 . This means that they end in 1.6 which uses an ZAP witness $((s_b, w_0), (s_{1-b}, w_1))$, which is the same as 0.6, only in reverse order.

Hybrids 0.6 and 1.6 are computationally indistinguishable. This follows directly from the computational hiding of the commitment $\vec{C}$ to b . Indeed, assume towards contradiction that $\mathcal{V}$ distinguishes the two hybrids. Concretely, denote the probability it outputs 1 on 0.6 by p_0 , and the probability it outputs 1 on 1.6 by p_1 , and assume WLOG that $p_0 - p_1 \geq \varepsilon(n)$, for some noticeable $\varepsilon(n)$. We can construct a predictor that given a commitment $\vec{C} = \mathcal{C}(b)$ to a random bit $b \leftarrow \{0, 1\}$, guesses b with probability $\frac{1+\varepsilon(n)}{2}$. The predictor, samples a random $b' \leftarrow \{0, 1\}$ as a candidate guess for b , and performs the experiment corresponding to 0.6, only that it locates w_0 and w_1 according to b' , rather than the unknown b . If the distinguisher outputs 1, the predictor guesses $b = b'$ and otherwise it guesses $b = 1 - b'$.

Conditioned on $b = b'$, $\mathcal{V}$ is experiencing 0.6, and thus the guess will be correct with probability p_0 ; conditioned on $b = 1 - b'$, $\mathcal{V}$ is experiencing 1.6, and the guess will be right with probability $1 - p_1$. So overall the guessing probability is $\frac{p_0}{2} + \frac{1-p_1}{2} \geq \frac{1}{2} + \frac{\varepsilon(n)}{2}$. This completes the proof that the protocol is WI.

It is left to show that the protocol is an argument of knowledge. Indeed, let $\mathcal{P}^*$ be any prover that convinces the honest verifier of accepting with noticeable probability $\varepsilon(n)$, then with probability at least $\varepsilon(n)/2$ over its first message, it holds with probability at least $\varepsilon(n)/2$ over the rest of the protocol that $\mathcal{P}^*$ convinces $\mathcal{V}$. Let us call such a prefix **good**. Now for any **good** prefix, we can consider the perfectly binding induced commitment to the bit b , and from the soundness of the ZAP, we get a circuit that with probability at least $\varepsilon(n)/2 - \text{negl}(n)$ produces an accepting commitment transcript for the plaintext s_{1-b} , and gives a valid witness $w \in \mathcal{R}_{\mathcal{L}}$, padded with $\text{PRG}(s_{1-b})$. This in particular, means that we can first sample a prefix (hope it is **good**), and then use the extraction guarantee of the commitment to learn s_{1-b} and $\text{PRG}(s_{1-b})$, and thus also the witness w . This completes the proof of Lemma 81. $\square$

2-message WI with instance-independent first message. We shall also make use of 2-message WI with instance-independent first message. Here there are two verifier and prover messages. Like in the three message definition the verifier message does not depend on the instance, but is allowed to depend on its length. In such a protocol, we only require soundness. ZAPs, for instance, satisfy this requirement, but we can also do with a privately verifiable protocol rather than a ZAP. (In fact, also in the above construction of 3-message WIPOKs with instance-independent first message, the ZAPs

can be replaced with any 2-message WI with instance-independent first message.)

4.5.3 1-Hop Homomorphic Encryption

A *1-hop homomorphic encryption scheme* [89] allows a pair of parties to securely evaluate a function as follows: the first party encrypts an input, the second party homomorphically evaluates a function on the ciphertext, and the first party decrypts the evaluation result. Such a scheme can be instantiated based on garbled-circuits and an appropriate 2-message oblivious transfer protocol, based on either Decision Diffie-Hellman or Quadratic Residuosity [5, 89, 115, 135, 152].

Definition 82. A scheme $(\text{Gen}, \text{Enc}, \text{Eval}, \text{Dec})$, where Gen, Eval are probabilistic and Enc, Dec are deterministic, is a *semantically-secure, circuit-private, 1-hop homomorphic encryption scheme* if it satisfies the following properties:

- **Perfect correctness:** For any $n \in \mathbb{N}$, $u \in \{0, 1\}^n$ and circuit C :

$$\Pr_{\substack{\text{sk} \leftarrow \text{Gen}(1^n) \\ c \leftarrow \text{Enc}_{\text{sk}}(u) \\ \text{Eval}}} \left[\begin{array}{l} \hat{c} \leftarrow \text{Eval}_{\text{sk}}(c, C) \\ \text{Dec}_{\text{sk}}(\hat{c}) = C(u) \end{array} \right] = 1 .$$

- **Semantic security:** For any polysize $\mathcal{A}$, $n \in \mathbb{N}$, and any pair of inputs $u_0, u_1 \in \{0, 1\}^n$

$$\Pr_{\substack{b \leftarrow \{0, 1\} \\ \text{sk} \leftarrow \text{Gen}(1^n)}} [\mathcal{A}(\text{Enc}_{\text{sk}}(u_b)) = b] < \frac{1}{2} + \text{negl}(n) .$$

- **Circuit privacy:** A randomized evaluation should not leak information on the input circuit C . This should hold even for malformed ciphertexts. Formally, let $\mathcal{E}(x) = \text{Supp}(\text{Enc}(x))$ be the set of all legal encryptions of x , let $\mathcal{E}_n = \cup_{x \in \{0, 1\}^n} \mathcal{E}(x)$ be the set legal encryptions for strings of length n , and let $\mathcal{C}_n$ be the set of all circuits on n input bits. There exists a (possibly unbounded) simulator $\mathcal{S}$ such that:

$$\begin{aligned} \{C, \text{Eval}(c, C)\}_{\substack{n \in \mathbb{N}, C \in \mathcal{C}_n \\ x \in \{0, 1\}^n, c \in \mathcal{E}(x)}} &\approx_c \{C, \mathcal{S}(c, C(x), |C|)\}_{\substack{n \in \mathbb{N}, C \in \mathcal{C}_n \\ x \in \{0, 1\}^n, c \in \mathcal{E}(x)}} \\ \{C, \text{Eval}(c, C)\}_{\substack{n \in \mathbb{N} \\ C \in \mathcal{C}_n, c \notin \mathcal{E}_n}} &\approx_c \{C, \mathcal{S}(c, \perp, |C|)\}_{\substack{n \in \mathbb{N} \\ C \in \mathcal{C}_n, c \notin \mathcal{E}_n}} . \end{aligned}$$

4.5.4 Constructions

In this section, we construct zero-knowledge protocols against verifiers with bounded advice from generalized extractable one-way functions against adversaries with bounded auxiliary input (GEOWFs against bounded auxiliary-input adversaries). We start by describing a construction of a 3-message argument of knowledge from any GEOWF, 1-hop homomorphic encryption, and 3-message WIPOK with instance-independent first message. We then show a 2-message argument, assuming (non-interactive) commitments that can be inverted in super-poly time $T(n)$, GEOWFs that are hard against $\text{poly}(T(n))$ -size adversaries, and any 2-message WI with instance-independent verifier message (in particular, ZAPs).

4.5.4.1 A 3-message zero-knowledge argument of knowledge

Let $(\text{Gen}, \text{Enc}, \text{Eval}, \text{Dec})$ be a semantically-secure, circuit-private, 1-hop homomorphic encryption scheme. Let (w_1, w_2, w_3) denote the messages of 3-message WIPOK with an instance-independent first message (as in Definition 80). Let $\mathcal{F} = \{f_n\}_{n \in \mathbb{N}}$ be a key-less GEOWF, against $(b(n) + 2n)$ -bounded-auxiliary-input adversaries, with respect to a privately-verifiable relation $\mathcal{R}^{\mathcal{F}} = \{\mathcal{R}_n^{\mathcal{F}}\}_{n \in \mathbb{N}}$. Let $\mathcal{T}(u, u')$ be the efficient tester for $\mathcal{R}_n^{\mathcal{F}}(f_n(u), u')$. We further denote by $\mathcal{T}_{v,u'}(u)$ a circuit that, given input u , verifies that “ $v \neq f_n(u)$ or $\mathcal{T}(u, u') = 1$ ”; that is, either “ u is not a valid preimage of v , or $\mathcal{R}_n^{\mathcal{F}}(f_n(u), u') = 1$ ”. Also, let $\mathbf{1}$ be a circuit of the same size as $\mathcal{T}_{v,u'}$ that always returns 1. The protocol is given in Figure 4.8.

Theorem 83. *Protocol 4.8 is a zero-knowledge argument of knowledge against b -bounded-auxiliary-input verifiers.*

High-level idea behind the proof. For simplicity let us explain why the protocol is sound, showing it is an argument of knowledge follows a similar reasoning. Assuming that $\varphi \notin \mathcal{L}$, in order to pass the WIPOK, with respect to an evaluated cipher $\hat{c}$ that decrypts to 1, the prover must know a witness u' such that $\mathcal{T}_{v,u'}(u) = 1$. This, by definition, and the fact that the verifier indeed sends an image v together with its encrypted preimage u , means that u' must be such that u' satisfies $\mathcal{R}^{\mathcal{F}}(f_n(u), u') = 1$, and thus the prover actually violates $\mathcal{R}^{\mathcal{F}}$ -hardness (formally, we also need to invoke semantic security to claim that the encryption of u does not help in producing such a witness.)

Protocol 4.8

Common Input: $\varphi \in \mathcal{L} \cap \{0, 1\}^n$.

Auxiliary Input to $\mathcal{P}$: a witness w for φ .

1. $\mathcal{P}$ sends the first message $w_1 \in \{0, 1\}^n$ of the instance-dependent WIPOK.
2. $\mathcal{V}$ samples $u \leftarrow \{0, 1\}^{\ell(n)}$ and $\text{sk} \leftarrow \text{Gen}(1^n)$, computes $v = f_n(u)$, $c_u = \text{Enc}_{\text{sk}}(u)$ and sends (v, c_u) , as well as the second WIPOK message w_2 .
3. $\mathcal{P}$ samples $\hat{c} \leftarrow \text{Eval}(\mathbf{1}, c_u)$, and sends $\hat{c}$, together with the WIPOK message w_3 stating that:

$$\{\varphi \in \mathcal{L}\} \bigvee \{\exists u' : \hat{c} = \text{Eval}(\mathcal{T}_{v, u'}, c_u)\} \quad ,$$

using the witness $w \in \mathcal{R}_{\mathcal{L}}(\varphi)$.

4. $\mathcal{V}$ verifies the proof and that $\text{Dec}_{\text{sk}}(\hat{c}) = 1$.

Figure 4.8: A 3-message ZK argument of knowledge against verifiers with b -bounded auxiliary-input.

To show ZK, we use the fact that if the verifier sends v together with an encryption of a true preimage u , the the simulator can invoke the extractor and extract a witness u' from its code and auxiliary input, and use it to complete the WIPOK. Here we use the bound on the first WI prover message, to claim that the overall auxiliary-input is appropriately bounded. In case, the verifier diverges from the protocol, and doesn't send proper v and encrypted preimage, the definition of $\mathcal{T}_{v, u'}$ guarantees that the circuit will also accept in this case. Thus in either case, the circuit privacy of homomorphic evaluation would guarantee indistinguishability from a real proof, where the prover actually evaluates the constant $\mathbf{1}$ circuit.

A more detailed proof follows.

Proof. We first show that the protocol is an argument of knowledge.

Claim 4.5.1. *Protocol 4.8 is an argument of knowledge against arbitrary poly-size provers.*

Proof. Let $\mathcal{P}^*$ be any polysize prover that convinces $\mathcal{V}$ of accepting with noticeable probability $\varepsilon(n)$. The witness extractor would derive from $\mathcal{P}^*$ a new prover for $\mathcal{P}_{\text{wi}}^*$ that emulates $\mathcal{P}^*$ in the WIPOK; in particular, it would honestly sample (v, c_u) as part of the second verifier message that $\mathcal{P}^*$ gets. The extractor would then choose the random coins r for $\mathcal{P}_{\text{wi}}^*$, sample a transcript ts of an execution with the honest WIPOK

verifier $\mathcal{V}_{\text{wi}}$, and apply the WIPOK extractor on the transcript ts , with oracle access to $\mathcal{P}_{\text{wi}}^*$. The WIPOK extractor then hopefully obtains a witness for the WI statement

$$\{\varphi \in \mathcal{L}\} \bigvee \{\exists u' : \hat{c} = \text{Eval}(\mathcal{T}_{v,u'}, c_u)\} ,$$

where (v, c_u) are those honestly sampled by $\mathcal{P}_{\text{wi}}^*$, and $\hat{c}$ is output by $\mathcal{P}^*$.

We claim that, with noticeable probability $\varepsilon(n)^2/2 - \text{negl}(n)$, we find a witness w for the first part of the statement $\varphi \in \mathcal{L}$. Otherwise, we can use $\mathcal{P}^*$ to break the $\mathcal{R}^{\mathcal{F}}$ -hardness of $\mathcal{F}$. To prove this claim, we first note that the emulated transcript ts in this experiment is distributed identically to the transcript in a real execution of $\mathcal{P}^*$ with the honest verifier. Thus, we know that such a transcript ts is accepted by $\mathcal{V}$ with probability at least $\varepsilon(n)$. Now, let us call random coins r for $\mathcal{P}_{\text{wi}}^*$ good if they are such that with probability at least $\varepsilon(n)/2$ over the coins of the WIPOK verifier $\mathcal{V}_{\text{wi}}$, it accepts the proof given by $\mathcal{P}_{\text{wi}}^*$. Since we know that overall $\mathcal{V}_{\text{wi}}$ accepts with probability at least $\varepsilon(n)$, then by a standard averaging argument, at least an $\varepsilon(n)/2$ fraction of the coins r for $\mathcal{P}_{\text{wi}}^*$ are good. Furthermore, conditioned on a transcript ts that is accepted by $\mathcal{V}$, the probability that the corresponding coins r are good increases. Thus, it follows that the probability that ts is accepting and the corresponding coins r are good is at least $\varepsilon(n) \cdot \varepsilon(n)/2$. Now, recall that, whenever this occurs, the extractor for the WIPOK would also output a witness for the corresponding statement (except with negligible probability).

We would like to show that the extracted witness is the one for the $\varphi \in \mathcal{L}$ statement. Indeed, assume that, with noticeable probability $\eta(n)$, it holds that ts is accepting, the extractor outputs a witness, but the witness is for the second statement. This, in particular, means that the witness extractor outputs u' such that $\hat{c} = \text{Eval}(\mathcal{T}_{v,u'}, c_u)$, where $\hat{c}$ is the output of $\mathcal{P}^*$. Moreover, since the transcript is accepted by $\mathcal{V}$, we know that $\text{Dec}(\hat{c}) = 1$. By correctness of decryption, this means that $\mathcal{T}_{v,u'}(u) = 1$, which in turn implies that $\mathcal{T}(u, u') = 1$, since $v = f_n(u)$. In other words, u' is a valid witness satisfying $\mathcal{R}_n^{\mathcal{F}}(f_n(u), u') = 1$.

We can now construct a breaker for the $\mathcal{R}^{\mathcal{F}}$ -hardness of $\mathcal{F}$. The breaker, given $v = f_n(u)$, would simply emulate all of the experiment above on its own, where $\mathcal{P}_{\text{wi}}$ would use v , and an encryption of zero $c_0 = \text{Enc}_{\text{sk}}(0)$ to emulate the second verifier message, instead of sampling $(v = f_n(u), c_u)$ on its own. We claim that it would obtain

the desired witness u' with noticeable probability $\eta(n) - \text{negl}(n)$. Indeed, had we used an encryption c_u of the preimage of v , instead of a zero-encryption, we know that it would produce a valid witness u with probability $\eta(n)$. Thus, the claim follows by the semantic security of Enc . This completes the proof of Claim 4.5.1 $\square$

We next show that the protocol is ZK. We note that, since the ZK simulator is allowed to simulate the (apriori unbounded) randomness of the verifier $\mathcal{V}^*$, we can restrict attention to verifiers $\mathcal{V}^*$ that only have bounded randomness. Indeed (assuming there exist OWFs), we can always consider a new verifier $\tilde{\mathcal{V}}^*$ that first stretches its bounded randomness using a PRG and then emulates $\mathcal{V}^*$. Then to simulate the view of $\mathcal{V}^*$, we can first apply the simulator $\tilde{\mathcal{S}}$ for $\tilde{\mathcal{V}}^*$, and then apply the PRG on the simulated randomness to obtain a full simulated view for $\mathcal{V}^*$. In particular, from hereon we can simply focus on deterministic verifiers $\mathcal{V}^*$ that get their bounded randomness as part of their bounded advice.

Claim 4.5.2. *Protocol 4.8 is ZK against any polytime verifier $\mathcal{V}^*$ with auxiliary-input of size at most $b(n)$.*

Proof. We describe a universal ZK simulator $\mathcal{S}$ and show its validity (universality is in the sense of Remark 79). Let $\varphi \in \mathcal{L}$ and let $\mathcal{V}^*$ be the code of any malicious verifier, and let z' be any advice of length at most $b(n)$. $\mathcal{S}$ starts by honestly computing the first message $w_{i1} \in \{0, 1\}^n$ of the WIPOK with instance-independent first message. It then feeds w_{i1} to $\mathcal{V}^*(\varphi; z')$ who returns (v, c, w_{i2}) that are (allegedly) an image under the function f_n , an encryption of a corresponding preimage, and the second message of the WIPOK.

$\mathcal{S}$ now constructs from the code of $\mathcal{V}^*$ a machine $\mathcal{M}_{\mathcal{V}^*}$ that, given 1^n and $z = (z', \varphi, w_{i1})$ as input, outputs some v , and whose running time is linear in the running time $t_{\mathcal{V}^*}$ of $\mathcal{V}^*$. Note that $|z| \leq |z'| + |\varphi| + |w_{i1}| \leq b(n) + 2n$, and thus, if $v = f_n(u)$ for some u , applying the extractor $\mathcal{E}$ on $\mathcal{M}_{\mathcal{V}^*}$ would result in a witness u' , such that $\mathcal{R}^{\mathcal{F}}(v, u') = 1$, in time $\text{poly}(t_{\mathcal{V}^*}^*)$. $\mathcal{S}$ does not test whether v is a valid image directly, it applies the extractor regardless to obtain a candidate u' , and then computes $\hat{c} = \text{Eval}(\mathcal{T}_{v, u'}, c)$. Then it sends $\hat{c}$ to $\mathcal{V}^*$, and completes the WIPOK using the trapdoor u' as a witness.

The validity of the simulator now follows by witness indistinguishability, as well as by the circuit privacy guarantee given by Eval . Specifically, we first move to a

hybrid simulator $\mathcal{S}'$ that proves the WIPOK statement using the actual witness w . The view generated by $\mathcal{S}'$ is indistinguishable from the one generated by $\mathcal{S}$ due to the WI property.

Now, we claim that the view generated by $\mathcal{S}'$ is indistinguishable from that generated by honest prover $\mathcal{P}$. First, note that the only difference between the two is that $\mathcal{P}$ sends $\hat{c} \leftarrow \text{Eval}(\mathbf{1}, c)$, whereas $\mathcal{S}'$ sends $\hat{c} \leftarrow \text{Eval}(\mathcal{T}_{v,u'}, c)$, for the extracted input u' . Now, note that if c is a valid ciphertext, then $\mathcal{T}_{v,u'}(\text{Dec}(c)) = \mathbf{1}(\text{Dec}(c)) = 1$; indeed, if $v = f_n(u)$ where $u = \text{Dec}(c)$, then the extracted u' is such that $\mathcal{T}(u, u') = 1$, and the above follows by the definition of $\mathcal{T}_{v,u'}(u)$. Thus, in this case, the distribution of $\hat{c}$ induced by $\mathcal{P}$ is indistinguishable from that induced by $\mathcal{S}'$, by circuit privacy. In fact, circuit privacy says that this is also the case if c is an invalid cipher. $\square$

This completes the proof of Theorem 83. $\square$

4.5.4.2 A 2-message zero-knowledge argument.

In this section, we show that, using complexity leveraging (and superpolynomial hardness assumptions), we can augment the protocol from the previous section to a 2-message argument.

Let $(\text{Gen}, \text{Enc}, \text{Eval}, \text{Dec})$ be a semantically-secure, circuit-private, 1-hop homomorphic encryption scheme. Let (w_1, w_2) denote the messages of 2-message WI with an instance-independent first message (as in Definition 80). Let $\mathcal{F} = \{f_n\}_{n \in \mathbb{N}}$ be a keyless GEOWF, against $(b(n) + n)$ -bounded-auxiliary-input adversaries, with respect to a privately-verifiable relation $\mathcal{R}^{\mathcal{F}} = \{\mathcal{R}_n^{\mathcal{F}}\}_{n \in \mathbb{N}}$. Further assume that $\mathcal{F}$ is one-way against adversaries of size $\text{poly}(T)$ (see Remark 75). Let $\mathcal{T}(u, u')$ be the efficient tester for $\mathcal{R}_n^{\mathcal{F}}(f_n(u), u')$. We further denote by $\mathcal{T}_{v,u'}(u)$ a circuit that, given input u , verifies that “ $v \neq f_n(u)$ or $\mathcal{T}(u, u') = 1$ ”; that is, either “ u is not a valid preimage of v , or $\mathcal{R}_n^{\mathcal{F}}(f_n(u), u') = 1$ ”. Also, let $\mathbf{1}$ be a circuit of the same size as $\mathcal{T}_{v,u'}$ that always returns 1. Let $\mathcal{C}$ be a perfectly binding commitment that is hiding against polysize adversaries, and can be completely inverted in time $T(n)$, for some computable super-polynomial function $T(n) = n^{\omega(1)}$. The protocol is given in Figure 4.9.

Theorem 84. *Protocol 4.9 is a zero-knowledge argument against b -bounded-auxiliary-input verifiers.*

Protocol 4.9

Common Input: $\varphi \in \mathcal{L} \cap \{0, 1\}^n$.

Auxiliary Input to $\mathcal{P}$: a witness w for φ .

1. $\mathcal{V}$ samples $u \leftarrow \{0, 1\}^{\ell(n)}$ and $\text{sk} \leftarrow \text{Gen}(1^n)$, computes $v = f_n(u)$, $\mathbf{c}_u = \text{Enc}_{\text{sk}}(u)$ and sends $(v, \mathbf{c}_u)$, as well as the first WI message wi_1 .
2. $\mathcal{P}$ samples a commitment to zero $C \leftarrow \mathcal{C}(0^\ell)$, $\hat{\mathbf{c}} \leftarrow \text{Eval}(\mathbf{1}, \mathbf{c}_u)$, and sends $(C, \hat{\mathbf{c}})$, together with the second WI message wi_2 stating that:

$$\{\varphi \in \mathcal{L}\} \bigvee \left\{ \exists u' : \begin{array}{l} \hat{\mathbf{c}} = \text{Eval}(\mathcal{T}_{v, u'}, \mathbf{c}_u) \\ C = \mathcal{C}(u') \end{array} \right\},$$

using the witness $w \in \mathcal{R}_{\mathcal{L}}(\varphi)$.

3. $\mathcal{V}$ verifies the proof and that $\text{Dec}_{\text{sk}}(\hat{\mathbf{c}}) = 1$.

Figure 4.9: A 2-message ZK argument against verifiers with b -bounded auxiliary input.

High-level idea behind the proof. Proving ZK against verifiers with bounded advice is essentially the same as in the 3-message protocol, only that now the simulator also commits to the input that it extracts from the verifier (and by the hiding of the commitment ZK is maintained). The proof of soundness is essentially the same as showing POK in the 3-message protocol, only that now, the WI does not provide witness extraction, instead we will extract a witness in time $\text{poly}(T(n))$, by inverting the prover's commitment with brute-force. Since one-wayness holds even against $\text{poly}(T(n))$ -adversaries, soundness follows.

A more detailed proof follows.

Proof sketch. We first show that the protocol is a sound against polysize adversaries.

Claim 4.5.3. *Protocol 4.9 is an argument.*

Proof sketch. Let $\mathcal{P}^*$ be any polysize prover, and assume towards contradiction that for infinitely many $\varphi \notin \mathcal{L}$, $\mathcal{P}^*$ convinces $\mathcal{V}$ of accepting with noticeable probability $\varepsilon(n)$. We show to break the $\mathcal{R}^{\mathcal{F}}$ -hardness of $\mathcal{F}$. The breaker, given v would sample a first WI message wi_1 , and encryption of zero $\mathbf{c}_0$, and feed $(v, \mathbf{c}_0, \text{wi}_1)$ to $\mathcal{P}^*$, who outputs a commitment C , an alleged image v , and a proof wi_2 for the statement

$$\{\varphi \in \mathcal{L}\} \bigvee \left\{ \exists u' : \begin{array}{l} \hat{\mathbf{c}} = \text{Eval}(\mathcal{T}_{v, u'}, \mathbf{c}_u) \\ C = \mathcal{C}(u') \end{array} \right\}.$$

By the semantic security of the 1-hop encryption, the above is indistinguishable from an experiment in which the breaker uses c_u for an actual preimage of v , and thus we know that with probability $\varepsilon(n) - \text{negl}(n)$ the proof is convincing. By the soundness of the WI system, and since $\varphi \notin \mathcal{L}$, it follows that C is a commitment to a proper witness u' . The inverter can now break C in time $T(n)$ and thus break $\mathcal{R}^{\mathcal{F}}$ -hardness of $\mathcal{F}$. $\square$

We next show that the protocol is ZK. As noted in the previous section, we can restrict attention to deterministic verifiers $\mathcal{V}^*$ that get their bounded randomness as part of their bounded advice.

Claim 4.5.4. *Protocol 4.9 is ZK against any polytime verifier $\mathcal{V}^*$ with advice of size at most $b(n)$.*

Proof sketch. We describe a universal ZK simulator $\mathcal{S}$ and show its validity (universality is in the sense of Remark 79). Let $\varphi \in \mathcal{L}$ and let $\mathcal{V}^*$ be the code of any malicious verifier, and let z' be any advice of length at most $b(n)$. $\mathcal{S}$ starts by running $\mathcal{V}^*(\varphi; z')$ who returns (v, c, wi_1) that are (allegedly) an image of the of the function f_n , an encryption of its preimage, and the verifier message of the WI protocol.

$\mathcal{S}$ now constructs from the code of $\mathcal{V}^*$ a machine $\mathcal{M}_{\mathcal{V}^*}$ that, given 1^n and $z = (z', \varphi)$ as input, outputs some v , and whose running time is linear in the running time $t_{\mathcal{V}^*}$ of $\mathcal{V}^*$. In particular, $|z| \leq |z'| + |\varphi| \leq b(n) + n$. $\mathcal{S}$ then applies the extractor $\mathcal{E}$ on $\mathcal{M}_{\mathcal{V}^*}$, and obtains a candidate witness $u' \in \{0, 1\}^\ell$ in time $\text{poly}(t_{\mathcal{V}^*})$.

$\mathcal{S}$ now computes $\hat{c} = \text{Eval}(\mathcal{T}_{v, u'}, c)$, as well as a commitment C to u' , and completes the WI using the trapdoor u' as a witness. It sends $(C, \hat{c}, \text{wi}_2)$ to complete the simulation.

The validity of the simulator now follows by witness indistinguishability, as well as the circuit privacy guarantee. Specifically, we can first move to a hybrid simulator $\mathcal{S}'$ that proves the WI statement using the witness w . The view generated by $\mathcal{S}'$ is indistinguishable from the one generated by $\mathcal{S}$ due to the WI property. Now, we can claim that the view generated by $\mathcal{S}'$ is indistinguishable from that generated by the honest prover $\mathcal{P}$. Indeed, the only difference between the two is that $\mathcal{P}$ commits to 0^ℓ instead of u' , and sends $\hat{c} \leftarrow \text{Eval}(\mathbf{1}, c)$, whereas $\mathcal{S}'$ sends $\hat{c} \leftarrow \text{Eval}(\mathcal{T}_{v, u'}, c)$, for the extracted input u' . The two views are indistinguishable by the hiding of the commitment and by the function privacy guarantee of the 1-hop evaluation (this is argued exactly as in the proof of Claim 4.5.2). $\square$

This completes the proof of Theorem 84. $\square$

4.6 Black-Box Lower Bounds

In our construction of EOWFs (or GEOWFs) against bounded-auxiliary-input adversaries, the extractor is non-black-box, i.e., it makes explicit use of the adversary's code. In particular, the simulation of our 2-message and 3-message ZK protocols, which invokes this extractor, makes a non-black-box use of the adversarial verifier. In this section, we show that this is inherent by extending known results for adversaries with unbounded polynomial advice to the case of bounded-advice adversaries. We also observe that such black-box impossibilities do not hold for totally uniform adversaries (having no advice at all on top of their constant size description).

EOWF with black-box extractors. We sketch why there do not exist EOWFs against b -bounded auxiliary-input adversaries where $b = n^{\Omega(1)}$, for security parameter n , and where the extractor only uses the adversary as a black-box (a similar implication can be shown for the case of generalized EOWFs). Specifically, we show that given a function family $\mathcal{F}$ that satisfies one-wayness, there does not exist a PPT black-box extractor $\mathcal{E}$ such that for any PPT adversary $\mathcal{M}$, any security parameter $n \in \mathbb{N}$, and any advice $z \in \{0, 1\}^{b(n)}$:

$$\Pr_{e \leftarrow \mathcal{K}_{\mathcal{F}}(1^n)} \left[\begin{array}{cc} v \leftarrow \mathcal{M}(e; z) & \wedge & u' \leftarrow \mathcal{E}^{\mathcal{M}(\cdot; z)}(e) \\ \exists u : f_e(u) = v & & f_e(u') \neq v \end{array} \right] \leq \text{negl}(n) .^4$$

This essentially follows the same idea behind the impossibility presented in Section 4.3, only that now some of the computation done there by the obfuscated auxiliary-input can be shifted from the auxiliary-input to the adversary itself, as it is anyhow accessed as a black-box. Concretely, consider the adversary $\mathcal{M}$ that interprets its auxiliary input z as a seed k of a pseudo-random function that maps the keys of $\mathcal{F}$ to inputs of $\mathcal{F}$. On input $(e; z)$, $\mathcal{M}$ computes an input $u = \text{PRF}_z(e)$ and outputs $v = f_e(u)$. Using the guarantee of the pseudo-random function, it is not hard to see that any black-box extractor $\mathcal{E}$ can be used to break the one-wayness property of $\mathcal{F}$ (using a much simplified version of the proof in Section 4.3).

Note that the above does not hold when $b(n) = O(\log(n))$, since then the advice

cannot contain a seed for a secure pseudo-random function. In fact, when $b(n) = O(\log(n))$, any family that is EOWF against b -bounded auxiliary-input adversaries also has a black-box extractor. The extractability property of the EOWF guarantees the existence of an extractor for every adversary $\mathcal{M}$ and advice z . Since there are only polynomially many different pairs $(\mathcal{M}, z)$, a black-box extractor can run the (possibly non-black-box) extractor for every such $(\mathcal{M}, z)$, and is guaranteed that one of these executions outputs a valid preimage.

3-round ZK with black-box simulation. Goldreich and Krawczyk [98] show that a 3-message protocol for a language $\mathcal{L} \notin \text{BPP}$ that is zero-knowledge against non-uniform verifiers cannot have a black-box simulator. That is, there is no simulator that only uses the verifier as a black-box. To show this, they first construct a specific family $\mathcal{V}$ of non-uniform verifiers, and then prove that any black-box simulator that can simulate verifiers in $\mathcal{V}$ can be used to decide $\mathcal{L}$ efficiently. This proof, however, does not directly rule out black-box simulation for bounded auxiliary-input verifiers. The reason is that, in the proof of [98], the advice given to verifiers in $\mathcal{V}$ encodes a key for a p -wise independent hash function where p bounds the running time of the simulator. Now, to rule out any polytime simulator, we must require simulation for verifiers with advice of arbitrary polynomial length.

However, assuming one-way functions exist, we can replace the p -wise independent hash function in the construction of $\mathcal{V}$ by a pseudo-random function with seed length that is independent of p . Then, using the same argument as [98], we can show that black-box simulation is impossible even for b -bounded auxiliary-input verifiers where $b(n) = n^{\Omega(1)}$.

Similarly to the case EOWF, there is no impossibility for 3-message ZK against b -bounded auxiliary-input verifiers where $b(n) = O(\log(n))$. In fact, as explained above, in this case, the non-black-box extractor of our GEOWF also implies a black-box extractor, which we can use to construct a black-box simulator in our 3-message ZK protocol.

2-round ZK. Goldreich and Oren [101] show that 2-message protocols for any language $\mathcal{L} \notin \text{BPP}$ that are zero-knowledge against non-uniform verifiers do not exist (even with non-black-box simulation). Their result crucially relies on the fact that the auxiliary-input of the verifier can encode the first message of the protocol

(and can in fact be extended to also rule out the case of bounded auxiliary-input verifiers, with advice longer than the first message). Our construction of 2-message ZK does not contradict the impossibility of [101] since it is only ZK against b -bounded auxiliary-input adversaries where b is smaller than the length of the first protocol message.

Chapter 5

SNARKs: When Knowledge Meets Succinctness

For the Snark's a peculiar creature, that won't
Be caught in a commonplace way.
Do all that you know, and try all that you don't:
Not a chance must be wasted to-day!

The Hunting of the Snark,
Lewis Carroll

This chapter presents our results on succinct non-interactive arguments of knowledge (SNARKs). The chapter is based on [28–30].

5.1 Overview

In this section, we give a high-level overview of the results in this chapter, and of the corresponding techniques. Before that, we recall the basic relevant concepts and main motivation, which were briefly described in the introduction.

5.1.1 Basic Concepts and Motivation

The notion of interactive proof systems [106] is central to both modern cryptography and complexity theory. One extensively studied aspect of interactive proof systems is their expressiveness; this study culminated with the celebrated result that $\text{IP} = \text{PSPACE}$ [148]. Another aspect of such systems is that proofs for rather complex **NP**-statements can potentially be verified much faster than by direct checking of an **NP** witness. This

aspect has received much attention in recent years due to the growing interest in the problem of verifiably delegating computation, and will be the aspect that we will focus on.

When requiring statistical soundness, significant savings in communication (let alone verification time) are unlikely [43, 96, 102, 150]. If we settle for proof systems with *computational* soundness, known as *argument systems* [49], then significant savings can be made. Using collision-resistant hashing (CRHs) and probabilistically-checkable proofs (PCPs) [9], Kilian [121] showed a four-message interactive argument for **NP** where, to prove membership of an instance x in a given **NP** language L with **NP** machine $\mathcal{M}$, communication and the verifier's time are bounded by $\text{poly}(n + |\mathcal{M}| + |x| + \log t)$, while the prover's running time by $\text{poly}(n + |\mathcal{M}| + |x| + t)$. Here t is the classical NP verification time of $\mathcal{M}$ for the instance x , n is a security parameter, and poly is a *universal* polynomial (i.e., independent of n , $\mathcal{M}$, x , and t). We call such argument systems *succinct*. (We note that, while traditionally the term succinctness referred only to communication complexity, we overload the term, requiring also fast verification.)

Non-interactive succinct arguments. Kilian's protocol requires four messages. A challenge, which is of both theoretical and practical interest, is removing interaction from succinct arguments. As a first step in this direction, Micali [128] constructed *one-message* succinct non-interactive arguments for **NP**, in the *random oracle model*, by applying the Fiat-Shamir paradigm [79] to Kilian's protocol.

In the plain model, it is known that one-message solutions are impossible for hard-enough languages (against non-uniform provers), so one usually considers the weaker goal of two-message succinct arguments where the verifier message is generated *independently* of the statement later chosen by the prover. Such arguments are called SNARGs. More precisely, a SNARG for a language $\mathcal{L}$ is a triple of algorithms $(\mathcal{G}, \mathcal{P}, \mathcal{V})$ where: (i) the generator $\mathcal{G}$, given the security parameter n , samples a *reference string* σ and a corresponding *verification state* τ ($\mathcal{G}$ can be thought to be run during an offline phase, by the verifier, or by someone the verifier trusts); (ii) the prover $\mathcal{P}(\sigma, x, w)$ produces a proof π for a statement " $x \in \mathcal{L}$ " given a witness w ; (iii) $\mathcal{V}(\tau, x, \pi)$ deterministically verifies the validity of π for that statement.¹

¹The term SNARG was coined in [91], there it only refers to the succinctness of the proof π itself; we overload the term requiring also fast verification time.

For the restricted case of verifying deterministic polynomial-time computations, there are several constructions relying on standard assumptions in various models [6, 27, 56, 59, 64, 65, 80, 84, 104, 118], most notably, a construction of a two-message argument for any language in $\mathbf{P}$ [119].

For $\mathbf{NP}$ computations the state of affairs is rather different. Indeed, Gentry and Wichs [91] show that, for hard-enough $\mathbf{NP}$ languages and adaptively chosen statements, no SNARG can be proven sound, in the plain model (rather than, say, the random oracle model) via a black-box reduction to a standard assumption (more accurately, any efficiently falsifiable assumption [134]). Early attempts to remove interaction from Kilian’s protocol by Aiello et al. [4] were shown to have inherent vulnerabilities [73], and in particular, could not be proven secure. Di Crescenzo and Lipmaa [71] show how to augment the approach of Aiello et al. to circumvent these vulnerabilities. They show that the construction is secure under an elaborate “knowledge assumption” that is intimately tied to their specific construction.² Another solution in the plain of the same spirit would be to assume that Micali’s CS proofs in the random oracle model can be imported to the plain model by instantiating the random oracle with a specific hash function (e.g., SHA) and assuming security.

Typically, we attempt to avoid relying on such assumptions, but rather rely on concise assumptions about each one of the primitives involved in the construction, which can then be investigated in isolation, independently of the construction. Thus the first question we focus on in this chapter is

Q 1: *Under what assumptions can we prove the existence of SNARGs?*

SNARGs with improved features. Another problem addressed in this chapter concerns certain qualitative feature of SNARGs. The first such feature is *public verifiability*. In a publicly-verifiable SNARG, the verification state τ , associated with the reference string σ , can be made public without compromising soundness. This is in contrast to a *designated-verifier* SNARG, where τ must be kept secret; in particular, τ must be protected from leakage, including the verifier’s responses when checking proofs. In publicly-verifiable SNARGs, leakage is not a concern, τ and σ can be used repeatedly, anyone who trusts the generation of τ can verify proofs, and proofs can be publicly archived for future use.

²The assumption, as formulated in [71], turns out to be false (see Section 5.1.3).

All previous SNARG constructions are of the designated-verifier kind (and there, indeed, an adversary learning the verifier’s responses on polynomially many proofs can break soundness). However, if one is willing to relax the succinctness requirement of SNARGs, and settle for a *preprocessing* SNARG, then the goal can be achieved. In such a SNARG, the verifier is allowed to conduct an expensive offline phase. More precisely, the generator $\mathcal{G}$ takes as additional input a time bound B , may run in time $\text{poly}(n + B)$ (rather than $\text{poly}(n + \log B)$), and generates a reference string σ and a verification state τ that can be used, respectively, to prove and verify correctness of computations of length at most B . A set of works [34, 85, 111, 125] achieves this weaker goal, commonly by using techniques that can be cast as a combination of *linear* PCPs and linearly homomorphic encryption/encoding with suitable knowledge properties [34]. We thus ask:

Q 2: *Can we construct publicly-verifiable SNARGs without preprocessing?*

Of course, we could always assume that Micali’s protocol, when the random oracle is instantiated with a sufficiently-complicated hash function, is sound and obtain a publicly-verifiable construction in the plain model. However, this assumption does not seem to be satisfying, because it strongly depends on the specific construction, and does not shed light on the required properties from such a hash function. Instead, we would like to have a solution whose soundness is based on a *concise* and *general* assumption that is “construction-independent” and can be studied separately.

Another feature of SNARGs, which we address, is the overhead they induce in prover complexity. While typically the focus is on minimizing the resources required by the verifier, minimizing those required by the prover is a critical goal, especially when considering the use of SNARGs for the goal of delegating computation. Indeed, the verifier is most likely to be *paying* to use the prover’s resources by renting servers from the cloud, and the more resources are used the greater the cost to the verifier. These resources include, not only time complexity, but also space complexity, which tends to be a severe problem in practice (often more so than time complexity).

When instantiating the PCP-based SNARG constructions (e.g., [71, 128]) with known time-efficient PCPs [24], the SNARG prover runs in time $t \cdot \text{poly}(n)$. However, the quasilinear running time of the prover is achieved via the use of FFT-like methods, which unfortunately demand $\Omega(t)$ space even when the computation of the **NP** verification machine $\mathcal{M}$ requires space s with $s \ll t$. The situation is even worse

in the preprocessing SNARGs of [34, 85, 111, 125], where the generator runs in time $\Omega(t) \cdot \text{poly}(n)$ to produce a reference string σ of length $\Omega(t) \cdot \text{poly}(n)$. This string must then be stored somewhere and accessed by the prover every time it proves a new statement; thus, once again, $\Omega(t)$ space is needed (in contrast to a SNARG without preprocessing where the generator runs in time $\text{poly}(n)$ and the reference string is short).

Ideally, we want SNARGs that simultaneously enable the verifier to run fast *and* enable the prover to use an amount of resources that is as close as possible to those required by the original computation. We thus define a *complexity-preserving* SNARG to be such that the prover runs in time $t \cdot \text{poly}(n)$ and space $s \cdot \text{poly}(n)$, and the verifier runs in time $|x| \cdot \text{poly}(n)$, when proving and verifying that a t -time computation, given by a (non-deterministic) machine $\mathcal{M}$ and input x . We ask:

Q 3: *Can we construct complexity-preserving SNARGs?*

SNARGs of knowledge - SNARKs. The standard soundness requirement from SNARGs is often insufficient. Indeed, in many settings, we would like to argue about problems for which the mere existence of a solution may be evident, but finding such a solution may be computationally hard. For instance, in the setting of delegation, the untrusted server may be required to perform a computation over a large authenticated database; here, it is not sufficient that the server simply proves that it used data for which *there exists* a valid signature, but rather it should prove that it *knows* such a signature. More generally, soundness alone will typically not suffice when composing SNARGs with other cryptographic primitives, or with themselves (as we shall see).

In such settings, it is essential that the SNARG is a SNARG of knowledge (SNARK) that also guarantees efficient witness extraction. Throughout, we will aim to achieve all of the goals stated above not only for SNARGs, but also for SNARKs.

5.1.2 Results

We make progress on the above three questions, showing new constructions of SNARGs and SNARKs, based on concise knowledge assumptions, and achieving new features.

We put forth the notion of *extractable collision-resistant hash functions* (ECRHs). We show that if ECRHs exist then so do SNARKs for **NP**.

Theorem 85 (informal). *Assuming extractable collision-resistant hash functions and private-information-retrieval schemes, there exist SNARKs for NP.*

We propose a candidate ECRH construction, based on a knowledge-of-exponent assumption and the hardness of discrete logs. We also define *generalized ECRHs* (akin to the generalized extractable one-way functions from Chapter 4). We show that these are sufficient for SNARKs, and in fact, also *necessary*.

Our second contribution is a *bootstrapping theorem for SNARKs*. Specifically, we show a transformation taking *any* SNARK, possibly with poor efficiency features (e.g., having expensive preprocessing, a prover running in quadratic time, a prover requiring large space, and so on), and outputs a SNARK, without expensive preprocessing, which is *complexity-preserving*. The transformation works in both publicly-verifiable and designated-verifier cases.

Theorem 86 (informal). *Assume existence of collision-resistant hash functions. Then any publicly-verifiable preprocessing SNARK can be efficiently transformed into a publicly-verifiable SNARK, without preprocessing, that is also complexity-preserving. Assuming in addition the existence of fully-homomorphic encryption, an analogous statement holds for the designated-verifier case.*

Applying our theorem to existing preprocessing SNARKs, based on variants of the knowledge-of exponent assumption [34, 85, 111, 125], we obtain positive answers for the aforementioned questions:

Corollary 87 (informal). *There exist publicly-verifiable complexity-preserving SNARKs, in the plain model, under knowledge-of-exponent assumptions.*

We note that the bootstrapping theorem not only implies an efficiency improvement for preprocessing SNARKs, but is also useful for improving the efficiency of SNARKs that do not necessarily have expensive preprocessing, yet are not complexity preserving.

Abstracting SNARK composition via proof-carrying data. Generalizing Valiant’s notion of incrementally-verifiable computation (IVC) [149], Chiesa and Tromer [57] study a more general security goal of *enforcing local properties* in dynamic distributed computations, while avoiding the communication and computation overhead of standard multi-party computation solutions. They introduce a primitive

called *proof-carrying data* (PCD), which allows to dynamically augment the messages of a given distributed computation (perhaps with no security guarantees) with short proofs attesting to the fact that the local property holds up to the current point. Using the PCD abstraction (and by choosing appropriate local properties to enforce), one can rather directly capture various scenarios which seem to require SNARK recursive composition, such as IVC and targeted malleability [40].

The SNARK bootstrapping techniques presented here lead PCD systems *in the plain model*, for large classes of distributed computation (including any sequential polytime computation). In [30], the PCD abstraction is also utilized to present the bootstrapping process itself. In the scope of this thesis, we will only present a bare-bone bootstrapping transformation for SNARK, without going through the PCD abstraction. For more details on PCDs, the reader is referred to [30].

Feasibility of implementations. An interesting feature of the SNARK resulting from our bootstrapping transformation, when starting from the protocols of [34, 85, 111, 125], is that it *does not invoke the PCP Theorem* but only elementary probabilistic-checking techniques (as interpreted in [34]). This provides an essentially different path to the construction of succinct arguments, deviating from the previous approaches, which are all PCP-based.

The fact that no heavy PCP machinery is involved, makes our SNARKs attractive for actual implementation. A first such implementation is presented by Ben-Sasson, Chiesa, Tromer and Virza [25].

We next elaborate on each of the results, and the techniques used to achieve it.

5.1.3 Designated-Verifier SNARKs from ECRHs

In this section, we define what are extractable collision-resistant hash functions, and sketch how they can be used, in conjunction with private-information-retrieval (PIR), to construct SNARKs.

At a high-level, our final construction follows an approach similar to that of Di Crescenzo and Lipmaa [71] meant to squash Kilian’s four-message succinct argument [121]. We thus start by reviewing the two.

Kilian’s Construction. Kilian constructs a succinct argument for $\mathbf{NP}$, $(\mathcal{P}, \mathcal{V})$, where the prover $\mathcal{P}$ uses a Merkle tree [127] in order to provide to $\mathcal{V}$ “virtual access” to a PCP proof.³ More precisely, to prove a statement $y \in \mathcal{L}$, the verifier $\mathcal{V}$ starts by sending the prover a key e for a collision-resistant hash function h_e . The prover, given a witness w , constructs a PCP-proof ξ . To guarantee efficient verification, $\mathcal{P}$ cannot send $\mathcal{V}$ the witness w nor ξ . Rather, $\mathcal{P}$ builds a Merkle hash tree over the proof ξ (using the hash function h_e). The hash value `root` corresponding to the root of the tree is sent to the verifier and is effectively a succinct commitment to ξ , which also supports succinct decommitment for any specific bit in ξ (by giving all the hash values along the path from that bit to the root, including all the siblings along the way; this is also known as an “authentication path”). The verifier $\mathcal{V}$ tosses random coins r for the PCP verifier and sends them to the prover. The prover $\mathcal{P}$ then sends back answers to those queries according to the proof ξ , together with the authentication paths. The verifier accepts if all the answers are consistent with the root value `root`, and convinces the PCP verifier.

At high-level, the soundness follows from the fact that the Merkle tree provides the verifier “virtual access” to the PCP proof, in the sense that given the root value of the Merkle tree, for every query q , it is infeasible for a computationally-bounded cheating prover to answer q differently depending on the queries. Therefore, interacting with the prover is “equivalent” to having access to a PCP proof oracle. Soundness can then be reduced to that of the PCP system.

The “PCP+MT+PIR approach”: Di Crescenzo and Lipmaa [71] propose to “squash” Kilian’s four-message protocol into a two-message protocol, with what we refer to as the PCP-Merkle-Tree-Private-Information-Retrieval (PCP+MT+PIR) approach. In Kilian’s protocol, the verifier obtains from the prover a Merkle hash to a PCP oracle and only then reveals its queries to the prover. In [71], the two verifier messages are combined into one message. Specifically, in the first message, the verifier sends, along with the hash key e , a PIR-encrypted version of the PCP queries (the first message of a PIR scheme can be viewed as an encryption to the queries); the prover then prepares the required PCP oracle ξ , computes a Merkle hash of ξ , and answers the verifier’s queries by replying to the PIR queries according to a database that contains

³Kilian’s argument is also zero-knowledge. Since our focus is only on soundness, we shall describe a simpler version of the protocol that is not zero-knowledge.

each bit of ξ together with its corresponding authentication path with respect to the Merkle hash).

The soundness of the scheme is proved based on the assumption that any convincing prover $\mathcal{P}^*$ must essentially behave as an honest prover; namely, if a proof is accepting, then the prover must have in mind a *full* PCP oracle ξ , which maps under the Merkle hash procedure to the claimed root, and such a proof ξ can be obtained by an efficient extractor $\mathcal{E}_{\mathcal{P}^*}$.⁴ It is then shown that, if this is the case, the extracted string ξ must be consistent with the answers the prover provides to the PCP queries, for otherwise the extractor can be used to obtain collisions of the hash function underlying the Merkle tree. Therefore, the extracted string ξ also passes the PCP test, where the queries are encrypted under PIR. Then, it follows from the privacy of the PIR scheme that, the string ξ is “computationally independent” of the query. Then, from the soundness of PCP, they conclude that the statement must be true.

Towards a more concise assumption - ECRHs. We obtain a SNARG (or in fact, a SNARK) based on a more restricted assumption of extractable collision-resistant hash functions, which we now formulated.

$\mathcal{H} = \{h_e\}$ is an extractable collision-resistant function family (ECRH) if it is

1. collision-resistant in the standard sense: no computationally-bounded adversary given a random key e , can produce a collision in h_e ,
2. extractable: for any efficient adversary $\mathcal{A}$, there is an extractor $\mathcal{E}_{\mathcal{A}}$, such that, if given a random key e , $\mathcal{A}(e)$ produces a valid image $v = h_e(u)$, $\mathcal{E}_{\mathcal{A}}(e)$ produces a corresponding preimage $u' \in h_e^{-1}(v)$, except with negligible probability.

For our construction, we will in fact need auxiliary-input extraction (as already discussed in Chapter 4); namely, extraction should hold even when $\mathcal{A}$ is given some auxiliary information z , which is also given to the extractor $\mathcal{E}_{\mathcal{A}}$. While in Chapter 4 we show that such a definition cannot hold with respect to arbitrary polynomial auxiliary input, for our construction, we can settle for a relaxed definition that only considers of a-priori bounded size, and which is taken from a specific distribution over auxiliary inputs. See further discussion in Remark 108.

⁴As originally formulated, the assumption of [71] turns out to be false; indeed, a malicious prover can always start from a good PCP oracle ξ for a true statement and compute an “almost full” Merkle hash on ξ , skipping very few branches — so one should at least formulate an analogous but more plausible assumption by only requiring “sufficient consistency” with the claimed root.

How to use ECRHs to construct SNARKs. At a very high-level, our construction follows the “PCP+MT+PIR approach” but replacing the collision-resistant hash function underlying the Merkle tree in their construction with an ECRH.

Unlike in [71] where “global extraction” is assumed directly, we show how to *lift* the “local extraction” guarantee provided by the ECRH, to the “global extraction” guarantee on the entire Merkle tree. More precisely, by relying on the (local) ECRH extraction guarantee, we show that for every convincing prover, there is an extractor that can efficiently extract a PCP proof $\tilde{\xi}$ that is “sufficiently satisfying” (i.e., a PCP verifier given $\tilde{\xi}$ accepts with high probability). Furthermore, to obtain the argument of knowledge property, we instantiate the underlying PCP system with PCPs of knowledge, which allow witness-extraction from any sufficiently-satisfying proof oracle. (See details for the requisite PCP system in Section 5.3.2.3.) We next describe the high-level ideas for achieving global extraction guarantees from the local extraction of ECRHs.

From local to global extraction. The main technical challenge lies in establishing a “global” extraction guarantee (namely, extraction of a sufficiently satisfying proof $\tilde{\xi}$ from any convincing prover) from the “local” extraction guarantee given by ECRHs.

A natural attempt is to start from the root of the Merkle tree and extract the values of the intermediate nodes of the Merkle tree layer by layer towards the leaves; that is, to extract a candidate proof $\tilde{\xi}$ by recursively applying the ECRH-extractor to extract the entire Merkle tree $\widetilde{\text{MT}}$, where the leaves should correspond to $\tilde{\xi}$.

However, recursively composing ECRH-extractors encounters a difficulty: each time applying the ECRH extraction incurs a polynomial blowup in extraction time. Therefore, (without making a very strong assumption on the amount of “blowup” incurred by the extractor,) we can only apply the ECRH extraction recursively a constant number of times; as a result, we can only extract from Merkle trees of a constant depth. We address this problem by opting to use a “squashed” Merkle tree, where the fan-in of each intermediate node is polynomial rather than binary as in the traditional case. Consequently, for every NP-statement, since its PCP proof is of polynomial length, the depth ℓ of the Merkle tree built over the PCP proof is a constant. Then, we can apply the ECRH extraction procedure recursively to extract the whole Merkle tree, while overall incurring polynomial overhead in the size of the extractor.

A technical problem that arises when applying the above recursive extraction pro-

cedure is that we might not be able to extract out a full Merkle tree (where all paths have the same length). More precisely, after applying the ECRH extraction recursively ℓ times, we obtain the values for the intermediate nodes up to level ℓ (thinking about the root as level zero). However, at each intermediate node, when applying the ECRH extractor, extraction could have failed to extract a preimage if the given intermediate node is not a valid hash image under the ECRH. Hence, the extracted tree might be at some points “disconnected”.⁵ Nevertheless, we show (relying solely on ECRH extraction) that the leaves in the extracted (perhaps partial) tree connected to the root are sufficiently satisfying for witness-extraction.

Proof at high level. Given the foregoing discussion, we show the correctness of the extraction procedure in two steps:

- *Step 1: “local consistency”.* We first show that whenever the verifier is convinced, the recursively extracted string $\tilde{\xi}$ contains valid answers to the verifier’s PCP queries specified in its PIR queries. Otherwise, it is possible to find collisions within the ECRH h as follows. A collision finder could simulate the PIR-encryption on its own, invoke both the extraction procedure and the prover, and obtain two paths that map to the same root but must differ somewhere (as one is satisfying and the other is not) and therefore obtain a collision.
- *Step 2: “from local to global consistency”.* Next, using the privacy guarantee of the PIR scheme, we show that, whenever we extract a set of leaves that are satisfying with respect to the PIR-encrypted queries, the same set of leaves must also be satisfying for almost all other possible PCP queries and is thus sufficient for witness-extraction. Indeed, if this was not the case then we would be able to use the polynomial-size extraction circuit to break the semantic security of the PIR.

For further details of the proof we refer the reader to Section 5.3.3.2.

5.1.4 Bootstrapping SNARKs

We now give the basic intuition behind our SNARK bootstrapping theorem. Specifically, we explain how to transform any SNARK $(\mathcal{G}, \mathcal{P}, \mathcal{V})$ (possibly very inefficient, and

⁵This captures for example the behavior of the prover violating the [71] assumption described above.

with expensive preprocessing) into a complexity-preserving SNARK $(\mathcal{G}^*, \mathcal{P}^*, \mathcal{V}^*)$ (that, in particular, has no expensive preprocessing). We focus on the case where $(\mathcal{G}, \mathcal{P}, \mathcal{V})$ is publicly-verifiable.

Suppose that $(\mathcal{G}, \mathcal{P}, \mathcal{V})$ is a preprocessing SNARK (this only makes the transformation harder because we must get rid of preprocessing). Recall that, in such a SNARK, the online verification phase is fast, but the offline phase is allowed to be expensive, in the following sense. The generator $\mathcal{G}$ takes as an additional input a time bound B , may run in time $\text{poly}(n + B)$, and generates a (potentially long) reference string σ and a (short) verification state τ that can be used, respectively, to prove and verify correctness of computations of length at most B . The (online) verifier $\mathcal{V}$ still runs in time $\text{poly}(n)$, independently of B . (Additionally, no guarantees are made about the time and space complexity of the honest prover, except that they are both bounded by $\text{poly}(n + B)$.) We would like to construct $(\mathcal{G}^*, \mathcal{P}^*, \mathcal{V}^*)$ so that $\mathcal{G}^*$ runs in time $\text{poly}(n)$ (which in particular bounds the size of the reference string σ) and $\mathcal{P}^*$ runs in time $t \cdot \text{poly}(n)$ and space $s \cdot \text{poly}(n)$.

At high-level, the idea is to first represent the long t -step computation of the machine $\mathcal{M}$ to be verified as a collection of $O(t)$ small $\text{poly}(n)$ -step computations, and then use recursive composition to aggregate many SNARK proofs for the correctness of each of these small computations into a *single* SNARK proof. Indeed, by repeatedly invoking the prover $\mathcal{P}$ of the “inefficient” preprocessing SNARK $(\mathcal{G}, \mathcal{P}, \mathcal{V})$ only to prove the correctness of *small computations*, we would “localize” the effect of the SNARK’s inefficiency. Specifically, when running the generator $\mathcal{G}$ in the offline phase in order to produce (σ, τ) , we would only have to budget for a time bound $B = \text{poly}(n)$, thereby making running $\mathcal{G}$ cheap. Furthermore, if the collection of small computations can be computed in time t and space s (up to $\text{poly}(n)$ factors), then running $\mathcal{P}$ on each of these small computations in order to produce the final proof would only take time $t \cdot \text{poly}(n)$ and space $s \cdot \text{poly}(n)$. Overall, we would achieve complexity preservation. Let us make this intuition somewhat more concrete.

Starting point: incrementally-verifiable computation. A natural starting point towards fulfilling the above plan is trying to use of the idea of incrementally-verifiable computation (IVC) [149]. Recall that the goal in IVC is to transform a given computation into a new computation that after every step outputs its current state and a proof

of its correctness so far, while preserving the time and space efficiency of the original computation (up to $\text{poly}(n)$ factors).

Specifically, to be convinced that there exists a witness w for which the machine $\mathcal{M}$ accepts w within t steps, it suffices to be convinced that there is a sequence of t' states $S_0, S_1, \dots, S_{t'}$ of $\mathcal{M}$ (with $t' \leq t$) that (a) starts from an initial state, (b) ends in an accepting state, and (c) every state correctly follows the previous one according to the transition function of $\mathcal{M}$.

This suggests the following solution, which we can think of as happening “in the mind” of the new SNARK prover $\mathcal{P}^*$. Using the SNARK prover $\mathcal{P}$, a first party P_1 proves to a second party P_2 that the state S_1 was generated by running the first step of $\mathcal{M}$ correctly. Then, again using $\mathcal{P}$, P_2 proves to a third party P_3 that, not only did he evaluate the second step of $\mathcal{M}$ correctly and obtained the state S_2 , but he *also* received state S_1 carrying a valid proof (i.e., accepted by the SNARK verifier $\mathcal{V}$) claiming that S_1 was generated correctly. Then, P_3 uses $\mathcal{P}$ to prove to the fourth party P_4 that, not only did he evaluate the third step of $\mathcal{M}$ correctly and obtained the state S_3 , but he *also* received state S_2 carrying a valid proof claiming that S_2 was generated correctly, and so on until the last party who, upon receiving a state carrying a proof of validity, proves that the last state is accepting. A verifier at the end of the chain gets a single, easy-to-verify, proof aggregating the correctness of all the steps in $\mathcal{M}$ ’s computation. Hopefully, by relying on the proof of knowledge property of the SNARK, it will be possible to recursively extract from any convincing prover a full transcript of the computation attesting to its correctness.

From the above IVC-based approach to our eventual goal of complexity-preserving SNARKs without preprocessing there is still a significant gap; we now describe the difficulties and how we overcome them.

Challenge 1: IVC with preprocessing SNARKs, and the size of local computations. In his construction of IVC, Valiant relies on the existence of publicly-verifiable SNARKs that *do not have expensive preprocessing*. In our setting, we have only a preprocessing SNARK at hand, so we have to ensure that each of the computations whose correctness we are proving is shorter than the time bound B associated with the preprocessing. Specifically, B must be larger than the running time of the SNARK verifier $\mathcal{V}$ plus a single computation step of $\mathcal{M}$. This is reminiscent of the *bootstrapping*

paradigm in fully-homomorphic encryption [88], where, in order to bootstrap a somewhat homomorphic scheme, homomorphic evaluation should support the decryption operation plus a single computation step. Whereas in bootstrapping of homomorphic encryption the challenge is to get the decryption circuit to be small enough, in our setting the running time of $\mathcal{V}$ (even for a preprocessing SNARK) is already $\text{poly}(n)$ -small, and the challenge is to get the computation required to perform one step of $\mathcal{M}$ to be small enough. Indeed, running step i in the “middle” of the computation requires computation proportional to size of the corresponding state S_i . Such a computations may thus be as large as the space s used by $\mathcal{M}$, which in turn could be as large as $\Omega(t)$. If, instead, we could ensure that each local computation being proven is of size $\text{poly}(n)$, then we could set $B = \text{poly}(n)$ and thereby avoid expensive preprocessing.

To achieve this goal, we invoke a “computational” reduction [24, 39] that transforms $\mathcal{M}$ into machine $\mathcal{M}'$ that requires only $\text{poly}(n)$ space and preserves the proof of knowledge property (i.e., any computationally-bounded adversary producing a witness that $\mathcal{M}'$ accepts can be efficiently used to find a witness that $\mathcal{M}$ accepts). The idea is that $\mathcal{M}'$ emulates $\mathcal{M}$ but does not bother to explicitly store its memory; instead, reads from memory are implemented non-deterministically, and their correctness is verified via dynamic Merkle hashing. These non-deterministic reads, and corresponding Merkle verification paths, can be appended to the witness for the computation, whose length, crucially, does not affect the time to run a step of the machine. (To ensure that the new computation steps are small enough, we also ensure that each step only looks at a small chunk of the witness).

This strategy also ensures that the resulting SNARK is complexity-preserving. Indeed, reducing $\mathcal{M}$ to the “small-space” $\mathcal{M}'$ and its representation as a $\tilde{O}(t)$ -step step-wise computation can be done “on the fly”, using the same time t and space s as the original computation, up to $\text{poly}(n)$ factors.

Challenge 2: extractor efficiency and the depth of the computation. As mentioned, to prove that the above approach is secure, we need to rely on proof of knowledge, in order to perform recursive extraction. This means that a proof of security based on recursive knowledge extraction will work for only a *constant* number of recursive compositions (due to the polynomial blowup in extractor size for each such composition). However, the step-wise computation we described has *polynomial*

depth. Valiant showed that, if the knowledge extractor is extremely efficient (linear in the prover’s size), then the problem can be avoided by aggregating proofs along a tree rather than along a path. We avoid Valiant’s assumption by extending his idea into aggregating proofs along “wide proof trees” of constant depth (similarly to the construction of SNARKs from extractable collision-resistant hash functions presented in the previous subsection.)

The case of designated-verifier SNARKs. So far we have assumed that the SNARK $(\mathcal{G}, \mathcal{P}, \mathcal{V})$ is publicly verifiable. What happens in the designated-verifier case? In this case, it is not clear how a party can prove that he verified a received proof without actually knowing the corresponding private verification state (which we cannot allow because doing so would void the security guarantee of the SNARK). We solve this problem by showing how to carefully use fully-homomorphic encryption to recursively compose proofs *without* relying on intermediate parties knowing the verification state.

5.1.5 More Related Work

We address several additional related works.

A prior (somewhat) succinct argument from a Knowledge of Exponent assumption. Mie [130] observes that the PCP+MT+PIR approach works as long as the PIR scheme is *database aware* — essentially, a prover that is able to provide valid answers to PIR queries must “know” their decrypted values, or, equivalently, must “know” a database consistent with those answers (by arbitrarily setting the rest of the database). Mie then shows how to make the PIR scheme of Gentry and Ramzan [90] PIR-aware, based on Damgård’s Knowledge of Exponent assumption [67]; unfortunately, while the communication complexity is very low, the receiver in [90], and thus also the verifier in [130], are inefficient relative to the database size.

Other SNARKs from ECRHs. Goldwasser, Lin, and Rubinfeld [105] consider a notion of ECRH similar to ours and construct (designated-verifier) SNARKs using a similar approach to ours. (A joint extended version of [105] and [29] that also includes applications of extractable one-way functions to zero-knowledge can be found in [28]).

Damgård et al. [68] consider the problem of succinct non-interactive secure computation. They also consider a notion of ECRH, which, unlike ours, requires seemingly

stronger interactive extractability assumptions; their notion does not seem to be implied by the existence of SNARKs.

Complexity-preserving SNARKs from extractable FHE. In [33], the MIP-based approach of Aiello et al. [4] is augmented so to yield a provably-secure SNARKs, assuming a strong form of fully-homomorphic encryption *with an extractable homomorphism property*. The SNARKs constructed in [33] are, in fact, complexity-preserving. However, that construction is for designated verifiers and also relies on a rather specific knowledge assumption.

Chapter Organization

In Section 5.2, we give the relevant definitions for SNARKs. In Section 5.3, we define ECRHs and GECRH, and present the corresponding construction of designated-verifier SNARKs. In Section 5.4, we present the SNARKs bootstrapping transformation.

5.2 Basic Definitions

The universal relation and NP restrictions. We define the *universal relation* $\mathcal{R}_{\mathcal{U}}$ [14], which provide a canonical representation for verification-of-computation problems. We also define **NP** restrictions of $\mathcal{R}_{\mathcal{U}}$.

Definition 88. *The **universal relation** is the set $\mathcal{R}_{\mathcal{U}}$ of instance-witness pairs $(y, w) = ((\mathcal{M}, x, t), w)$, where $\mathcal{M}$ is a Turing machine that accepts (x, w) after at most t steps. We denote by $\mathcal{L}_{\mathcal{U}}$ the **universal language** corresponding to $\mathcal{R}_{\mathcal{U}}$.*

*For any $c \in \mathbb{N}$, $\mathcal{R}_c \subseteq \mathcal{R}_{\mathcal{U}}$ is an **NP restriction** of $\mathcal{R}_{\mathcal{U}}$, consisting of those pairs $(y, w) = ((\mathcal{M}, x, t), w)$ for which $t \leq |x|^c$. Denote by $\mathcal{L}_c$ the corresponding language.*

5.2.1 SNARKs

A *succinct non-interactive argument of knowledge* (SNARK) for **NP** is a triple of algorithms $(\mathcal{G}, \mathcal{P}, \mathcal{V})$ that works as follows. The generator $\mathcal{G}$, on input the security parameter n and a time bound B , samples a *reference string* σ and a corresponding *verification state* τ . The honest prover $\mathcal{P}(\sigma, y, w)$ produces a proof π for the statement $y = (\mathcal{M}, x, t) \in \mathcal{R}_c$ given a valid witness w , provided that $t \leq B$; then $\mathcal{V}(c, \tau, y, \pi)$ verifies π . We shall use the notation $\mathcal{V}_c(\tau, y, \pi)$ for $\mathcal{V}(c, \tau, y, \pi)$.

The SNARK is *fully-succinct* if $\mathcal{G}$ runs “fast”, almost independently of B ; otherwise, it is of the *preprocessing* kind.

Definition 89. A triple of algorithms $(\mathcal{G}, \mathcal{P}, \mathcal{V})$, where $\mathcal{G}$ is probabilistic and $\mathcal{P}, \mathcal{V}$ are deterministic, is a **SNARK** for **NP** if the following holds:

1. Completeness

For every security parameter $n \in \mathbb{N}$, every time bound $B \in \mathbb{N}$, and every instance-witness pair $(y, w) = ((\mathcal{M}, x, t), w) \in \mathcal{R}_c$ with $t \leq |x|^c \leq B$,

$$\Pr \left[\mathcal{V}_c(\tau, y, \pi) = 1 \mid \begin{array}{l} (\sigma, \tau) \leftarrow \mathcal{G}(1^n, B) \\ \pi \leftarrow \mathcal{P}(\sigma, y, w) \end{array} \right] = 1 .$$

2. Adaptive proof of knowledge

For every $c \in \mathbb{N}$, and every polynomial-size prover $\mathcal{P}^*$ there exists a polynomial-size extractor $\mathcal{E}_{\mathcal{P}^*}$ such that for every security parameter $n \in \mathbb{N}$, every auxiliary input $z \in \{0, 1\}^{\text{poly}(n)}$, and every time bound $B \in \mathbb{N}$,

$$\Pr \left[\begin{array}{l} \mathcal{V}_c(\tau, y, \pi) = 1 \\ (y, w) \notin \mathcal{R}_c \end{array} \mid \begin{array}{l} (\sigma, \tau) \leftarrow \mathcal{G}(1^n, B) \\ (y, \pi) \leftarrow \mathcal{P}^*(z, \sigma) \\ w \leftarrow \mathcal{E}_{\mathcal{P}^*}(z, \sigma) \end{array} \right] \leq \text{negl}(n) .$$

3. Efficiency

There exists a universal polynomial p such that, for every $c \in \mathbb{N}$, for every large enough security parameter $n \in \mathbb{N}$, every time bound $B \in \mathbb{N}$, and every instance $y = (\mathcal{M}, x, t)$ with $t \leq B$,

- $\mathcal{G}(1^n, B)$ runs in time $\begin{cases} p(n) & \text{for fully-succinct SNARKs} \\ p(n + B) & \text{for preprocessing SNARKs} \end{cases}$;
- $\mathcal{P}(\sigma, y, w)$ runs in time $\begin{cases} p(n + |\mathcal{M}| + |x| + t) & \text{for fully-succinct SNARKs} \\ p(n + |\mathcal{M}| + |x| + B) & \text{for preprocessing SNARKs} \end{cases}$;
- $\mathcal{V}_c(\tau, y, \pi)$ runs in time $p(n + |\mathcal{M}| + |x|)$;
- an honestly generated proof has size $p(n)$.

We distinguish between the case where the proof of knowledge guarantee holds even if the verification state τ is made public to the case that τ needs to remain private.

Definition 90 (public vs. designated verification). *A SNARK is **publicly-verifiable** (a *pvSNARK*) if proof of knowledge holds when the verification state τ appears in the clear as part of the reference string σ , given to the prover. Otherwise, the SNARK is **designated-verifier** (a *dvSNARK*).*

We make several remarks regarding the above definitions.

Remark 91 (Universal SNARKs vs. SNARKs for **NP**). *The SNARKs given in Definition 89 are for **NP** relations $\mathcal{R}_c \subseteq \mathcal{R}_{\mathcal{U}}$. We stress that for such SNARKs, while the proof of knowledge guarantee only holds with respect to the **NP** relation $\mathcal{R}_c \subset \mathcal{R}_{\mathcal{U}}$, the polynomial p governing the efficiency of the SNARK is still required to be universal and independent of c .*

Remark 92 (The extractor $\mathcal{E}$ and auxiliary input). *Above, we require that any polynomial-size family of circuits $\mathcal{P}^*$ has a specific polynomial-size family of extractors $\mathcal{E}_{\mathcal{P}^*}$; in particular, we allow the extractor to be of arbitrary polynomial-size and to be “more non-uniform” than $\mathcal{P}^*$ (this is referred to as individual auxiliary-input in Remark 60). In addition, we require that for any prover auxiliary input $z \in \{0, 1\}^{\text{poly}(n)}$, the polynomial-size extractor manages to perform its witness-extraction task given the same auxiliary input z (this is referred to as common auxiliary-input in Remark 60). This aspect of the definition is, in fact, too strong assuming indistinguishability obfuscation. Indeed, as we shall see SNARKs imply (generalized) extractable one-way functions, and are thus subject to the same limitations presented in Section 4.3.*

The definition can be naturally relaxed to consider only specific distributions of auxiliary inputs according to the required application, as well as a-priori bounded auxiliary input, which are not subject to any known limitations.

Remark 93 (Turing machines vs. RAMs). *For simplicity of exposition, in the above definitions, and throughout this chapter, we model the computations to be verified via Turing machines. However, all of the results presented also carry over to more realistic models of computation such as random-access machines. See [30] for details.*

5.3 Extractable Collision-Resistant Hash Functions and Designated-Verifier SNARKs

In this section, we define extractable collision-resistant hash functions, discuss constructions, and show how they can be used to construct designated-verifier SNARKs.

5.3.1 ECRHs

In this section, we define extractable collision-resistant hash functions (ECRHs) and generalized extractable collision-resistant hash functions (GECRHs).

Definition 94 (ECRH). *Let $m, \ell' < \ell$, be polynomially bounded length functions. An efficiently computable family of functions*

$$\mathcal{H} = \left\{ h_e : \{0, 1\}^{\ell(n)} \rightarrow \{0, 1\}^{\ell'(n)} \mid e \in \{0, 1\}^{m(n)}, n \in \mathbb{N} \right\} ,$$

associated with an efficient (probabilistic) key sampler $\mathcal{K}_{\mathcal{H}}$, is an ECRH if it is:

1. **Collision-resistant:** *For any polysize $\mathcal{A}$, and security parameter $n \in \mathbb{N}$:*

$$\Pr_{e \leftarrow \mathcal{K}_{\mathcal{H}}(1^n)} \left[\begin{array}{l} u, u' \leftarrow \mathcal{A}(e) \\ h_e(u') = h_e(u) \end{array} \right] \leq \text{negl}(n) .$$

2. **Extractable:** *For any polysize adversary $\mathcal{A}$, there exists a polysize extractor $\mathcal{E}$ such that, for any polynomial b , security parameter $n \in \mathbb{N}$, and $z \in \{0, 1\}^{b(n)}$:*

$$\Pr_{e \leftarrow \mathcal{K}_{\mathcal{H}}(1^n)} \left[\begin{array}{l} v \leftarrow \mathcal{A}(e; z) \quad \wedge \quad u' \leftarrow \mathcal{E}(e; z) \\ \exists u : h_e(u) = v \quad \quad h_e(u') \neq v \end{array} \right] \leq \text{negl}(n) .$$

Remark 95 (Image verification). *Similarly to the case of EOWFs (see Remark 57), one can require that $\mathcal{H}$ has an extra image verification property. Our main ECRH-based construction (presented in Section 5.3.3) does not require any verifiability features.*

Remark 96 (Auxiliary input). *Above, we require that any polynomial-size family of circuits $\mathcal{A}$ has a specific polynomial-size family of extractors $\mathcal{E}$; in particular, we allow the extractor to be of arbitrary polynomial-size and to be “more non-uniform” than $\mathcal{A}$ (this is referred to as individual auxiliary-input in Remark 60). In addition, we require*

that for any auxiliary input $z \in \{0, 1\}^{\text{poly}(n)}$, the polynomial-size extractor manages to perform its preimage-extraction task given the same auxiliary input z (this is referred to as common auxiliary-input in Remark 60). This aspect of the definition is too strong assuming indistinguishability obfuscation. Indeed, ECRHs are, in particular, EOWFs, and are thus subject to the same limitations presented in Section 4.3.

While for presentational purposes the above definition may be simple and convenient, for our main application (i.e., SNARKs), we can actually settle for definition that is weaker—it can be restricted to a specific “benign distribution” on auxiliary inputs, which is also of a-priori bounded polynomial size (similarly to bounded auxiliary input EOWFs in Definition 58). In this relaxed setting, no negative results are known. (Naturally, when relying on ECRHs with respect to such restricted auxiliary-input distributions, the resulting SNARKs will account for the same auxiliary-input distributions.) See also Remark 108.

5.3.1.1 Generalized Extractable Collision-Resistant Hash Functions

We define *Generalized ECRHs* (GECRHs) akin to GEOWFs (Definition 61), which allow to express the tradeoff between extractability and different forms of collision-resistance in a flexible way. Concretely, the family $\mathcal{H}$ is now associated with a relation $\mathcal{R}_e^{\mathcal{H}}(h_e(u), u')$ on image-witness pairs $(v, u') \in \{0, 1\}^\ell \times \{0, 1\}^{\ell+k}$. Given a random key e , it is infeasible to find an input $u \in \{0, 1\}^\ell$ and witness $u' \in \{0, 1\}^{\ell+k}$, such that $u \parallel 0^k \neq u'$ and $\mathcal{R}_e^{\mathcal{H}}(h_e(u), u') = 1$. In contrast, a non-black-box extractor that is given a code that produces an image $h_e(u)$, for $u \in \{0, 1\}^\ell$, can find a witness $u' \in \{0, 1\}^{\ell+k}$, such that $\mathcal{R}_e^{\mathcal{H}}(h_e(u), u') = 1$.

As for GEOWFs, we consider two variants of GECRHs: The first is *publicly-verifiable GECRHs*, where for $(v = f_e(u), u')$, the relation $\mathcal{R}_e^{\mathcal{F}}(v, u')$, can be efficiently tested given v and u' only (and the key e if the function is keyed). The second is *privately-verifiable GEOWFs*, where the relation $\mathcal{R}_e^{\mathcal{F}}(v, u')$, might not be efficiently testable given only $(v = f_e(u), u')$, but it is possible to efficiently test the relation given u in addition.

We note that standard ECRHs, as given in Definition 94 are, in particular, GEOWFs, where the relation $\mathcal{R}_e^{\mathcal{H}}(v, u')$ is such that $k = 0$, and simply tests whether $v = h_e(u')$.

Definition 97 (GECRHs). *An efficiently computable family of functions*

$$\mathcal{H} = \left\{ h_e : \{0, 1\}^{\ell(n)} \rightarrow \{0, 1\}^{\ell'(n)} \mid e \in \{0, 1\}^{m(n)}, n \in \mathbb{N} \right\} ,$$

associated with an efficient (probabilistic) key sampler $\mathcal{K}_{\mathcal{H}}$, is a GECRH, with respect to a relation $\mathcal{R}_e^{\mathcal{H}}(v, u')$ on triples $(e, v, u') \in \{0, 1\}^{m(n)+\ell'(n)+\ell(n)+k(n)}$, if it is:

1. **$\mathcal{R}^{\mathcal{H}}$ -Collision-resistant:** For any polysize $\mathcal{A}$ and security parameter $n \in \mathbb{N}$:

$$\Pr_{e \leftarrow \mathcal{K}_{\mathcal{H}}(1^n)} \left[\begin{array}{l} u, u' \leftarrow \mathcal{A}(e) \\ u' \neq u \mid 0^{k(n)} \\ \mathcal{R}_e^{\mathcal{H}}(h_e(u), u') = 1 \end{array} \right] \leq \text{negl}(n) .$$

2. **$\mathcal{R}^{\mathcal{H}}$ -Extractable:** For any polysize adversary $\mathcal{A}$, there exists a polysize extractor $\mathcal{E}$ such that, for any polynomial b , security parameter $n \in \mathbb{N}$, and $z \in \{0, 1\}^{b(n)}$:

$$\Pr_{e \leftarrow \mathcal{K}_{\mathcal{H}}(1^n)} \left[\begin{array}{l} v \leftarrow \mathcal{A}(e; z) \quad \wedge \quad u' \leftarrow \mathcal{E}(e; z) \\ \exists u : h_e(u) = v \quad \mathcal{R}_e^{\mathcal{H}}(v, u') \neq 1 \end{array} \right] \leq \text{negl}(n) .$$

We further say that the function is

- **Publicly-verifiable** if $\mathcal{R}_e^{\mathcal{H}}(h_e(u), u')$ can always be efficiently computed by a tester $\mathcal{T}(e, h_e(u), u')$.
- **Privately-verifiable** if $\mathcal{R}_e^{\mathcal{H}}(h_e(u), u')$ can be efficiently computed by a tester a tester $\mathcal{T}(e, u, u')$.

Remark 98 (Does $\mathcal{R}^{\mathcal{H}}$ -collision-resistance imply (standard) collision-resistance?). In principle, $\mathcal{R}^{\mathcal{H}}$ -hardness may not imply collision-resistance of $\mathcal{H}$. Although this is not needed for our purposes, we may further require that the relation $\mathcal{R}^{\mathcal{H}}$ includes all pairs $(h_e(u), u)$, and thus ensure that $\mathcal{R}^{\mathcal{H}}$ -collision-resistance does imply collision-resistance.

Remark 99 (Is any GECRHs also a GEOWF?). As mentioned before any (standard) ECRH is a EOWF, which follows directly from the known fact that any CRH is a OWF. A similar argument shows that any GECRHs is also a GEOWFs (with the same verification features). In particular, the limitations we show in Section 4.3 on GEOWFs apply to proximity GECRHs as well.

Remark 100 (GECRHs vs. Proximity ECRHs). In [29], a different variant of ECRHs called proximity ECRHs is defined. There a proximity relation $\sim$ is defined on the range of the function. Collision-resistant is strengthened to require that not only is finding a collision $u \neq u', h_e(u) = h_e(u')$ hard, but also finding u' such that $h_e(u) \sim h_e(u')$ is hard. Extractability is weakened so that the extractor only has to work when the adversary outputs v that is the image of some $u \in \{0, 1\}^\ell$, and is allowed to output u' from an extended domain $\{0, 1\}^{\ell+k}$ such that $h_e(u) \sim h_e(u')$ is hard. GECRHs allow the relation to be even more general. Any proximity ECRH with relation $\sim$ implies a GECRH with relation $\mathcal{R}$, such that $\mathcal{R}(h_e(u), u') = 1$ iff $h_e(u) \sim h_e(u')$.

5.3.1.2 ECRH Constructions

In this section we discuss a candidate construction for an ECRH, based on a Knowledge of Exponent assumption and the hardness of discrete logs, and a general construction of a GECRHs from any SNARK.

A candidate ECRH from t -KEA. The *Knowledge of Exponent assumption* (KEA) [67] states that any adversary that, given a generator g and a random element g^α in some appropriate group, manages to produce $g^x, g^{\alpha x}$, must “know” the exponent x . The assumption was later extended in [22, 113], by requiring that given $g^{r_1}, g^{r_1\alpha}, g^{r_2}, g^{r_2\alpha}$ it is infeasible to produce f, f^α without “knowing” x_1, x_2 such that $f = g^{x_1 r_1} g^{x_2 r_2} = g^{x_1 r_1 + x_2 r_2}$. The t -KEA assumption is a natural extension to $t = \text{poly}(n)$ pairs $g^{r_i}, g^{\alpha r_i}$.

Assumption 101 (t -KEA). *There exists an efficiently-samplable ensemble $\mathcal{G} = \{\mathcal{G}_n\}$ where each $(\mathbb{G}, g) \in \mathcal{G}_n$ consists of a group of prime order $p \in (2^{n-1}, 2^n)$ and a generator $g \in \mathbb{G}$, such that the following holds. For any polysize adversary $\mathcal{A}$ there exists a polynomial-size extractor $\mathcal{E}_{\mathcal{A}}$ such that for any polynomial b , any $n \in \mathbb{N}$, and any auxiliary input $z \in \{0, 1\}^{b(n)}$,*

$$\Pr_{\substack{(\mathbb{G}, g) \leftarrow \mathcal{G}_n \\ (\alpha, \mathbf{r}) \leftarrow \mathbb{Z}_p \times \mathbb{Z}_p^t}} \left[\begin{array}{c} (f, f') \leftarrow \mathcal{A}(g^{\mathbf{r}}, g^{\alpha \mathbf{r}}, z) \\ f' = f^\alpha \end{array} \wedge \begin{array}{c} \mathbf{x} \leftarrow \mathcal{E}_{\mathcal{A}}(g^{\mathbf{r}}, g^{\alpha \mathbf{r}}, z) \\ g^{\langle \mathbf{x}, \mathbf{r} \rangle} \neq f \end{array} \right] \leq \text{negl}(n) ,$$

where $|\mathbb{G}| = p$, $\mathbf{r} = (r_1, \dots, r_t)$, $g^{\mathbf{r}} = (g^{r_1}, \dots, g^{r_t})$, $\mathbf{x} = (x_1, \dots, x_t)$, and $\langle \cdot, \cdot \rangle$ denotes inner product.

A related assumption was made by Groth [111]; there, instead of random $r_1, \dots, r_t$,

the exponents are powers of the same random element, i.e., $r_i = r^i$. (As formalized in [111], the assumption does not account for auxiliary inputs, but it could naturally be strengthened to do so.)

Our assumption can be viewed as a simplified version of Groth's assumption; in particular, we could use Groth's assumption directly to get ECRHs. Furthermore, Groth's assumption is formally stated in bilinear groups, while in our setting bilinearity is not necessary. When considered in (non bilinear) groups where t -DDH is assumed to hold, the two assumptions are actually equivalent.⁶ Therefore, as Groth shows that his assumption holds in the generic group model (independently of the bilinear structure) and as t -DDH is also known to hold in this model, our assumption holds in the generic group model as well.

Remark 102 (Auxiliary input). *When stated for arbitrarily long auxiliary-input the above assumption is too strong, assuming indistinguishability obfuscation; we can naturally restrict it to only consider auxiliary input of an a-priori bounded polynomial size $b(n)$, or to specific distributions. (See also Remark 96.)*

We now describe a construction of an ECRH $\mathcal{H}$ compressing $n \cdot t(n)$ bits to $2n$ bits:

- The key sampler $\mathcal{K}_{\mathcal{H}}(1^{secp})$: samples $(\mathbb{G}, g) \leftarrow \mathcal{G}_n$ and $(\alpha, \mathbf{r}) \xleftarrow{U} \mathbb{Z}_p \times \mathbb{Z}_p^t$, and output $e := (\mathbb{G}, g^{\mathbf{r}}, g^{\alpha \mathbf{r}})$.
- To compute $h_e(x_1, \dots, x_t)$: output the pair $(g^{\langle \mathbf{r}, \mathbf{x} \rangle}, g^{\langle \alpha \mathbf{r}, \mathbf{x} \rangle}) = \left(\prod_{i \in [t]} g^{r_i x_i}, \prod_{i \in [t]} g^{\alpha r_i x_i} \right)$.

The extractability of $\mathcal{H}$ easily follows from the t -KEA assumption. We show that $\mathcal{H}$ is collision-resistant based on the hardness of computing discrete logarithms in $\mathcal{G}$.

Claim 5.3.1. *If $\mathcal{C}$ finds a collision within $\mathcal{H}$ w.p. ε , then we can compute discrete logarithms with probability ε/t .*

Proof sketch. Given g^r , where $r \leftarrow \mathbb{Z}_p$, choose a random $i \in [t]$ and sample $\alpha, r_1, \dots, r_{i-1}, r_{i+1}, \dots, r_t$. Denote $r_i = r$ and $\mathbf{r} = (r_1, \dots, r_t)$. Feed $\mathcal{C}$ with $g^{\mathbf{r}}, g^{\alpha \mathbf{r}}$. By our initial assumption and the independent choice of i , $\mathcal{C}$ outputs $\mathbf{x}, \mathbf{x}'$ such that $x_i \neq x'_i$ and $g^{\langle \mathbf{x}, \mathbf{r} \rangle} = g^{\langle \mathbf{x}', \mathbf{r} \rangle}$ with probability at least ε/t . It follows that $r_i = (x_i - x'_i)^{-1} \sum_{j \in [n] - \{i\}} (x_j - x'_j) r_j$. $\square$

⁶ t -DDH asserts that, over suitable groups, tuples of the form $g^x, g^{x^2}, \dots, g^{x^t}$ are indistinguishable from random tuples.

From SNARKs to GECRHs. We show that SNARKs imply a natural construction of GECRHs; that is, GECRHs are not only sufficient, but also necessary, for SNARKs.

Proposition 103. *If there exist (even designated-verifier, and even preprocessing) SNARKs for $\mathbf{NP}$ and (standard) CRHs, then there exist a generalized GECRH $\mathcal{H}$ with a publicly-verifiable relation $\mathcal{R}^{\mathcal{H}}$.*

Proof sketch. Let $\mathcal{H}$ be a CRH that compresses $c(n) \cdot n$ bits to n bits, where $c(n) > 2$ is a compression parameter. Let $(\mathcal{P}, \mathcal{G}, \mathcal{V})$ be a SNARK for $\mathbf{NP}$ such that, given security parameter $\hat{n}$, the length of any proof is bounded by $\hat{n}^d$. We define an ECRH compressing $c \cdot n$ bits to $2n$ bits, $\tilde{\mathcal{H}}$. Any function is associated with a key $\tilde{e}$ sampled as follows:

1. draw a key for the CRH $e \leftarrow \mathcal{H}$,
2. draw public and private parameters $(\sigma, \tau) \leftarrow \mathcal{G}(n^{1/d})$, and
3. set $\tilde{e} = (e, \sigma)$.

Then, for an input u and letting $v = h_e(u)$, we define $h_{\tilde{e}}(u) = (v, \pi)$ where $\pi = \mathcal{P}(\sigma, \text{thm}, x)$ is a proof of knowledge for the $\mathbf{NP}$ -statement

$$\text{thm} := \text{“there exists } u \in \{0, 1\}^{c \cdot n} \text{ such that } h_e(u) = v \text{”} .$$

We now show that the above is a GECRH with respect to the relation $\mathcal{R}^{\tilde{\mathcal{H}}}((v, \pi), u')$ on $\{0, 1\}^{2n} \times \{0, 1\}^{c(n) \cdot n}$, that returns 1 iff $\exists u : h_e(u) = v$.

The generalized collision-resistance of $\tilde{\mathcal{H}}$ follows directly from the collision-resistance of $\mathcal{H}$, because any colliding pair (u, u') under $\mathcal{R}^{\tilde{\mathcal{H}}}$ is a colliding pair for $\mathcal{H}$. The generalized extractability property of $\tilde{\mathcal{H}}$ follows from the (adaptive) proof of knowledge of the SNARK $(\mathcal{P}, \mathcal{G}, \mathcal{V})$; that is, for any image-computing polysize adversary $\mathcal{A}$, the ECRH extractor is set to be the SNARK witness-extractor $\mathcal{E}_{\mathcal{A}}$. (We note that, for the proposition to go through, it is crucial for the SNARK to hold against adaptive provers; indeed, the adversary gets to choose on which inputs to compute the hash function, and these may very well depend on the public parameters.) $\square$

Why GECRH and not ECRH? At first glance, it is not clear why the above does not imply an (“exact”) ECRH rather than a GECRH. The reason lies in the fact that the extractor is only guaranteed to output one of many possible witnesses (preimages).

In particular, given an honest image $(h_e(u), \pi_u)$ (corresponding to some preimage u), the extractor may output u' such that $h_e(u') = h_e(u)$ but applying the honest prover to u' (or, more precisely, the NP-statement proving knowledge of u') results in a proof $\pi_{u'} \neq \pi_u$.

5.3.2 Preliminaries for SNARK Construction

In addition to ECRHs, we shall rely on several other basic building blocks, which we now overview.

5.3.2.1 Merkle Trees

Merkle tree (MT) hashing [127] enables a party to use a CRH to compute a succinct commitment to a long string ξ and later to locally open to any bit of ξ (again in a succinct manner). Specifically, given a function $h_e: \{0, 1\}^{\ell(n)} \rightarrow \{0, 1\}^n$ randomly drawn from a CRH ensemble, the committer divides ξ into $|\xi|/\ell(n)$ parts (padding with 0's if needed) and evaluates h_e on each of these; the same operation is applied to the resulting string, and so on, until one reaches the single n -bit root. For $|\xi| = (\ell/n)^{d+1}$, this results in a tree of depth d , whose nodes are all the intermediate n -bit hash images. An decommitment to a leaf in ξ (or any bit within it) includes all the nodes and their siblings along the path from the root to the leaf, and is of size ℓd . Typically, $\ell(n) = 2n$, resulting in a binary tree of depth $\log |\xi|$. In this work, we shall also be interested in “wide trees” with polynomial fan-in (relying on CRHs with polynomial compression); see Section 5.3.3.1.

5.3.2.2 Private Information Retrieval

A (single-server) polylogarithmic *private information retrieval* (PIR) scheme [50] consists of a triple of algorithms (PEnc, PEval, PDec) where:

- PEnc $_R(1^n, i)$ outputs an encryption C_i of query i to a database DB using randomness R ,
- PEval(DB, C_i) outputs a succinct blob e_i “containing” the answer DB[i], and
- PDec $_R(e_i)$ decrypts the blob e_i to an answer DB[i].

Formally:

Definition 104. A triple of algorithms $(\text{PEnc}, \text{PEval}, \text{PDec})$ is a PIR if it has the following properties:

1. **Correctness.** For any database DB , any query $i \in \{1, \dots, |\text{DB}|\}$, and security parameter $n \in \mathbb{N}$,

$$\Pr_R \left[\text{PDec}_R(e_i) = \text{DB}[i] : \begin{array}{l} C_i \leftarrow \text{Enc}_R(1^n, i) \\ e_i \leftarrow \text{PEval}(\text{DB}, C_i) \end{array} \right] = 1 ,$$

where $\text{PEval}(\text{DB}, C_i)$ runs in $\text{poly}(n, |\text{DB}|)$ time.

2. **Succinctness.** The running time of both $\text{PEnc}_R(1^n, i)$ and $\text{PEval}(\text{DB}, C_i)$ is bounded by

$$\text{poly}(n, \log |\text{DB}|) .$$

In particular, the sizes of the two messages C_i and e_i are also so bounded.

3. **Semantic security.** The query encryption is semantically secure for multiple queries, i.e., for any polynomial-size $\mathcal{A}$, security parameter $n \in \mathbb{N}$, and any two tuples of queries $\mathbf{i} = (i_1 \dots i_q), \mathbf{i}' = (i'_1 \dots i'_q) \in \{0, 1\}^{\text{poly}(n)}$,

$$\Pr \left[\mathcal{A}(\text{Enc}_R(1^n, \mathbf{i})) = 1 \right] - \Pr \left[\mathcal{A}(\text{Enc}_R(1^n, \mathbf{i}')) = 1 \right] \leq \text{negl}(n) ,$$

where $\text{Enc}_R(1^n, \mathbf{i})$ is the coordinate-wise encryption of the tuple $\mathbf{i}$.

PIR schemes with the above properties have been constructed under various hardness assumptions such as ΦHA [50] or LWE [48].

A-priori unknown database size. In our application, we will need the PIR server to be able to specify the database DB only *after* receiving a (random) query. The client will not know the database's size when generating his query, but will only know some superpolynomial bound, such as $|\text{DB}| \leq 2^{\log^2 n}$. We will require that the PIR scheme allows the server to deduce from an encrypted (long) query $r \in \{0, 1\}^{\log^2 n}$ a shorter ρ -bit query $\hat{r} \in \{0, 1\}^\rho$ that works for $|\text{DB}| = 2^\rho$, and that $\hat{r}$ is uniformly random when r is is.

This can be done by simply emulating $\log^2 n$ independent instances of any PIR scheme, where for a random $r \leftarrow \{0, 1\}^{\log^2 n}$, the receiver encrypts each one of its $\log^2 n$ prefixes. The sender then chooses the right one, according to the database size.

5.3.2.3 Probabilistically Checkable Proofs of Knowledge

A (verifier-efficient) *probabilistically checkable proof* (PCP) of knowledge for the universal relation $\mathcal{R}_{\mathcal{U}}$ is a triple of algorithms $(P_{\text{pcp}}, V_{\text{pcp}}, E_{\text{pcp}})$, where P_{pcp} is the prover, V_{pcp} is the (randomized) verifier, and E_{pcp} is the knowledge extractor.

Given $(y, w) \in \mathcal{R}_{\mathcal{U}}$, $P_{\text{pcp}}(y, w)$ generates a proof ξ of length $\text{poly}(t)$ and runs in time $\text{poly}(|y|, t)$. The verifier $V_{\text{pcp}}^{\xi}(y, r)$ runs in time $\text{poly}(|y|) = \text{poly}(|M| + |x| + \log t)$, while accessing $\text{polylog}(t)$ locations in the proof ξ and using $\rho = O(\log t)$ random bits. We require:

1. **Completeness.** For every $(y, w) = ((M, x, t), w) \in \mathcal{R}_{\mathcal{U}}$, $\xi \leftarrow P_{\text{pcp}}(y, w)$:

$$\Pr_{r \leftarrow \{0,1\}^{\rho(t)}} \left[V_{\text{pcp}}^{\xi}(y, r) = 1 \right] = 1 .$$

2. **Proof of knowledge (POK).** There is a constant ε such that for any $y = (M, x, t)$ if

$$\Pr_{r \leftarrow \{0,1\}^{\rho(t)}} \left[V_{\text{pcp}}^{\xi}(y, r) = 1 \right] \geq 1 - \varepsilon ,$$

then $E_{\text{pcp}}(y, \xi)$ outputs a witness w such that $(y, w) \in \mathcal{R}_{\mathcal{U}}$, and runs in time $\text{poly}(|y|, t)$.

(Note that proof of knowledge in particular implies that the soundness error is at most ε .)

PCPs of knowledge as defined above can be based on the efficient-verifier PCPs of [19, 26]. (See [149] for a simple example of how a PCP of proximity can yield a PCP with a proof of knowledge.) Moreover, the latter PCPs' proof length is quasi-linear in t ; for simplicity, we shall settle for a bound of t^2 .

We shall typically invoke the verifier V_{pcp} for $q(n)$ times to reduce the proof-of-knowledge threshold to $(1 - \varepsilon)^{q(n)}$, where n is the security parameter and $q(n) = \omega(\log n)$. Namely, extraction should succeed whenever $\Pr_{\mathbf{r}} \left[V_{\text{pcp}}^{\xi}(y, \mathbf{r}) = 1 \right] \geq (1 - \varepsilon)^q$, where $\mathbf{r} = (r_i)_{i \in [q]}$ and $V_{\text{pcp}}^{\xi}(y, \mathbf{r}) = \bigwedge_{i \in [q]} V_{\text{pcp}}^{\xi}(y, r_i)$.

5.3.3 Construction of SNARKs from ECRHs

In this section, we describe and analyze our construction of adaptive SNARKs for **NP** based on ECRHs, and then explain how the same construction can rely on GECRH, by appropriately augmenting the analysis.

Our modified approach. We modify the PCP+MT+PIR approach of [71] and show that the knowledge assumption of [71] (which involves the entire PIR+MT structure) can be replaced by the simpler generic assumption that ECRHs exist. Furthermore, our modification enables us to improve the result from a two-message succinct argument with non-adaptive soundness to an adaptive SNARG of knowledge (SNARK) — this improvement is crucial cryptographic applications. Specifically, we perform two modifications:

1. We instantiate the Merkle tree hash using an ECRH and, unlike the traditional construction where a CRH compressing $2n$ bits to n bits is used, we use a polynomially-compressing ECRH compressing n^2 bits to n bits; in particular, for n^{d+1} -long strings the resulting tree will be of depth d (rather than $d \log n$).⁷ As we shall see later, doing so allows us to avoid superpolynomial blowup of the final knowledge extractor that will be built via composition of ECRH extractors.
2. In order to ensure that the first message of the verifier does not depend on the theorem being proved, the database that we use does not consist of (authenticated) bits of ξ ; rather, the r -th entry of the database corresponds to the authenticated answers to the queries of $V_{\text{pcp}}^\xi(y, r)$ where y is chosen by the prover and, of course, the authentication is relative to a single string ξ (to avoid cheating provers claiming one value for a particular location of ξ in one entry of the database, and another value for the same location of ξ in another entry of the database). An additional requirement for this part is the use of a PIR scheme that can support databases where the exact size is a-priori unknown (see Section 5.3.2.2).

5.3.3.1 Construction Details

We start by providing a short description of our MT and then present the detailed construction of the protocol in Figure 5.1.

The shallow Merkle tree. By padding when required, we assume without loss of generality that the compressed string ξ is of size n^{d+1} (where d is typically unknown to the verifier). A node in the MT of distance j from the root can be represented by a string $\mathbf{i} = i_1 \cdots i_j \in [n]^j$ containing the path traversal indices (and the root is

⁷We note that any $(n^\varepsilon, n^{\varepsilon'})$ -compressing ECRH would have sufficed (for any constants $\varepsilon > \varepsilon'$); for the sake of simplicity, we stick with (n^2, n) -compression.

represented by the empty string). We then label the nodes with n -bit strings according to ξ and the hash $h_e : \{0, 1\}^{n^2} \rightarrow \{0, 1\}^n$ as follows:

- The leaf associated with $\mathbf{i} = i_1 \cdots i_d \in [n]^d \cong [n^d]$ is labeled by the $\mathbf{i}$ -th n -bit block of ξ , denoted by $\ell_{\mathbf{i}}$ (here $\mathbf{i}$ is interpreted as number in $[n^d]$).
- An internal node associated with $\mathbf{i} = i_1 \cdots i_j \in [n]^j$ is labeled by $h_e(\ell_{i_1} \ell_{i_2} \cdots \ell_{i_n})$, denoted by $\ell_{\mathbf{i}}$.
- Thus, the label of the root is $h_e(\ell_1 \ell_2 \cdots \ell_n)$, which we denote by $\ell_{\emptyset} = \text{MT}_e$.

The MT commitment is the pair $(d, \ell_{\emptyset})$. A decommitment $\mathbf{dcom}_{\mathbf{i}}$ to a leaf $\mathbf{i}$ consists of all the labels of all the nodes and their siblings along the corresponding path. To verify the decommitment information, evaluate the hash h from the leaves upwards. Specifically, for each node $\mathbf{i}' = \mathbf{i}j$ along the decommitment path labeled by $\ell_{\mathbf{i}'} = \ell_{\mathbf{i}j}$ and his siblings' labels $\ell_{i_1}, \ell_{i_2}, \dots, \ell_{i_{(j-1)}}, \ell_{i_{(j+1)}}, \dots, \ell_{i_n}$, verify that $h_e(\ell_{i_1}, \dots, \ell_{i_n}) = \ell_{\mathbf{i}}$.

We prove:

Theorem 105. *The construction in Figure 5.1 yields a (fully succinct) designated-verifier SNARK for NP.*

The completeness of the construction follows directly from the completeness of the PCP and PIR. In Section 5.3.3.2, we give a security reduction establishing the POK property (against adaptive provers).

5.3.3.2 Proof of Security

A high-level overview of the proof and main technical challenges are presented in Section 5.1.3. We now turn to the detailed proof, which concentrates on establishing and proving the validity of the knowledge extractor.

Proposition 106 (adaptive proof of knowledge). *For any $c \in \mathbb{N}$, and any polynomial-size $\mathcal{P}^*$ there exists a polynomial-size extractor $\mathcal{E}_{\mathcal{P}^*}$, such that for all $n \in \mathbb{N}$ and any auxiliary input $z \in \{0, 1\}^{\text{poly}(n)}$:*

$$\Pr_{e, \mathbf{r}, R} \left[\begin{array}{l} (y, \pi) \leftarrow \mathcal{P}^*(z, e, \text{PEnc}_R(\mathbf{r})) \quad \wedge \quad (y, w) \leftarrow \mathcal{E}_{\mathcal{P}^*}(1^n, z, e, \text{PEnc}_R(\mathbf{r})) \\ \mathcal{V}_c((1^n, e, R, \mathbf{r}), y, \pi) = 1 \quad \quad \quad w \notin \mathcal{R}_c(y) \end{array} \right] \leq \text{negl}(n) ,$$

where $e \leftarrow \mathcal{K}_{\mathcal{H}}(1^n)$ is a key for an ECRH, $\mathbf{r}$ are the PCP coins and R are the PIR coins.

Ingredients.

- A universal efficient-verifier PCP of knowledge $(P_{\text{pcp}}, V_{\text{pcp}}, E_{\text{pcp}})$ for $\mathcal{R}_{\mathcal{U}}$; for $((\mathcal{M}, x, t), w) \in \mathcal{R}_{\mathcal{U}}$, a proof ξ is s.t. $|\xi| \leq t^2$ and the non-repeated verifier uses $\rho = O(\log t)$ coins and $\text{polylog}(t)$ queries.
- A succinct PIR (PEnc, PEval, PDec) supporting an a-priori unknown database size.^a
- An ECRH compressing n^2 bits to n bits.

Setup $\mathcal{G}(1^n)$.

- Generate private verification state:
 - Sample coins for $q = \omega(\log n)$ repetitions of V_{pcp} : $\mathbf{r} = (r_1, \dots, r_q) \xleftarrow{U} \{0, 1\}^{(\log^2 n) \times q}$.
 - Sample coins for encrypting q PIR-queries: $R \xleftarrow{U} \{0, 1\}^{\text{poly}(n)}$.
 - Sample an ECRH key: $e \leftarrow \mathcal{K}_{\mathcal{H}}(1^n)$.
 - Set $\tau := (e, \mathbf{r}, R)$.
- Generate corresponding verifier-generated reference string:
 - Compute $C_{\mathbf{r}} \leftarrow \text{PEnc}_R(1^n, \mathbf{r})$.
 - Set $\sigma := (e, C_{\mathbf{r}})$.

Proof generation by $\mathcal{P}$.

- Input: $1^n, \sigma, (y, w) \in \mathcal{R}_c$ where $y = (\mathcal{M}, x, t)$ and $t \leq |x|^c$.
- Proof generation:
 - Compute a PCP proof $\xi \leftarrow P_{\text{pcp}}(y, w)$ of size $|\xi| = n^{d+1} \leq t^2$.
 - Compute an MT commitment for ξ : $\ell_\theta = \text{MT}_e(\xi)$ of depth d .
 - Let $\rho = O(\log t) < \log^2 n$ be the amount of coins required by V_{pcp} . Compute a database DB with 2^ρ entries; in each entry $\hat{r} \in \{0, 1\}^\rho$ store the decommitments $\text{dcom}_{\hat{r}}$ for all the locations of ξ that are queried by $V_{\text{pcp}}^\xi(y, \hat{r})$.^b
 - Compute $C_{\text{dcom}_{\hat{r}}} \leftarrow \text{PEval}(\text{DB}, C_{\mathbf{r}})$. Here each coordinate $r_j \in \{0, 1\}^{\log^2 n}$ of $\mathbf{r}$ is interpreted by the PIR as a shorter query $\hat{r}_j \in \{0, 1\}^\rho$.
 - The proof is set to be $\pi := (d, \ell_\theta, C_{\text{dcom}_{\hat{r}}})$.

Proof verification by $\mathcal{V}$.

- Input: $c, 1^n, \tau, y, \pi$, where $y = (\mathcal{M}, x, t), \pi = (d, \ell_\theta, C_{\text{dcom}_{\hat{r}}})$.
- Proof verification:
 - Verify that $n^{d+1} \leq t^2 \leq |x|^{2c}$.^c
 - Decrypt PIR answers $\text{dcom}_{\hat{r}} = \text{PDec}_R(C_{\text{dcom}_{\hat{r}}})$, and verify opened paths (against h_e and ℓ_θ).
 - Let $\xi|_{\hat{r}}$ be the opened values of ξ in the locations corresponding to $\hat{r}$ (where again $\hat{r}$ is the interpretation of $r \in \{0, 1\}^{\log^2 n}$ as $\hat{r} \in \{0, 1\}^\rho$). Check whether $V_{\text{pcp}}^{\xi|_{\hat{r}}}(y, \hat{r})$ accepts.
 - In case any of the above fail, reject.

^aSuch a PIR interprets a “long” query $r \in \{0, 1\}^{\log^2 n}$ as a shorter one $\hat{r} \in \{0, 1\}^\rho$, where $\rho \leq \log^2 n$, and $\hat{r}$ is the ρ -bit prefix of r (see Section 5.3.2.2).

^b V_{pcp} ’s queries might be adaptive; such behavior can be simulated by the prover.

^cThis is the single place where the verification algorithm depends on c .

Figure 5.1: A SNARK for NP.

Proof. Before describing the extraction procedure behind Proposition 106, we state a useful lemma concerning multiple-preimage extraction. Specifically, our extraction procedure will be based on recursive extraction along a Merkle tree structures; we will be interested in adversaries producing a vector of images $\vec{v}$. Here it will be convenient to invoke an extractor that can extract a vector of preimages $\vec{u}$ containing a preimage for each valid image in $\vec{v}$. This notion of extraction follows rather directly from the “single-instance” extraction notion of Definition 94.

Lemma 107 (Vector extraction). *Let $\mathcal{H}$ be an ECRH (as in Definition 94). Then for any polysize $\mathcal{A}$ there exists a polysize extractor $\mathcal{E}_{\mathcal{A}}^{\mathcal{H}}$ such that, for any polynomial b , security parameter $n \in \mathbb{N}$, and $z \in \{0, 1\}^{b(n)}$:*

$$\Pr_{e \leftarrow \mathcal{K}_{\mathcal{H}}(1^n)} \left[\begin{array}{l} \exists i : \quad \vec{v} \leftarrow \mathcal{A}(e; z) \quad \wedge \quad \vec{u}' \leftarrow \mathcal{E}_{\mathcal{A}}^{\mathcal{H}}(e; z) \\ \exists u : h_e(u) = \vec{v}_i \quad \quad \quad h_e(\vec{u}'_i) \neq \vec{v}_i \end{array} \right] \leq \text{negl}(n) .$$

We now turn to describing the extraction procedure and then prove that it satisfies Proposition 106. In order to simplify notation, we will address provers $\mathcal{P}^*$ that get as input only $(e, C_{\mathbf{r}})$, where $C_{\mathbf{r}} = \text{PEnc}_R(\mathbf{r})$; the analysis can be extended to the case that $\mathcal{P}^*$ also gets additional auxiliary input $z \in \{0, 1\}^{\text{poly}(n)}$.

The extraction procedure. As outlined in Section 5.1.3, extraction is done in two phases: first, we recursively extract along all the paths of the Merkle tree (MT); doing so results in a string (of leaf labels) $\tilde{\xi}$; then, we apply to $\tilde{\xi}$ the PCP witness-extractor E_{pcp} . As we shall see, $\tilde{\xi}$ will exceed the knowledge-threshold ε of the PCP and hence E_{pcp} will produce a valid witness.

We now describe the recursive extraction procedure of the ECRH-based MT. Given a polysize prover $\mathcal{P}^*$, let d be such that $|\mathcal{P}^*| \leq n^d$. We derive $2cd$ circuit families of extractors, one for each potential level of the MT. Define $\mathcal{E}_1 := \mathcal{E}_{\mathcal{P}^*}^{\mathcal{H}}$ to be the ECRH extractor for $\mathcal{P}^*$; like $\mathcal{P}^*$, $\mathcal{E}_1$ also expects input $(e, C_{\mathbf{r}}) \in \{0, 1\}^{\text{poly}(n)}$ and returns a string $(\tilde{\ell}_1, \dots, \tilde{\ell}_n) \in \{0, 1\}^{n \times n}$ (which will be a preimage in case $\mathcal{P}^*$ produces a valid image). We can interpret the string output by $\mathcal{E}_1$ as a vector of n elements in the range of the ECRH. We then rely on the vector extractor given by Lemma 107, and define $\mathcal{E}_2 := \mathcal{E}_{\mathcal{E}_1}^{\mathcal{H}}$ to be the extractor for $\mathcal{E}_1$. In general, we define $\mathcal{E}_{j+1} := \mathcal{E}_{\mathcal{E}_j}^{\mathcal{H}}$ to be the extractor for $\mathcal{E}_j$, where $\mathcal{E}_{j+1}$ expects input $(e, C_{\mathbf{r}})$ of the same format as $\mathcal{E}_j$. The extractor $\mathcal{E}_{j+1}(h, C_{\mathbf{r}})$

extracts a vector $(\tilde{\ell}_{\mathbf{i}k} : \mathbf{i} \in [n]^j, k \in [n]) \in \{0, 1\}^{n \times n \times n^j}$. Throughout, we will denote by $\mathcal{E}_{j+1}(h, C_{\mathbf{r}})[\mathbf{i}]$ the corresponding $(\tilde{\ell}_{\mathbf{i}1}, \dots, \tilde{\ell}_{\mathbf{i}n})$.

Recall that for any $\mathbf{i} \in [n]^j$ and $k \in [n]$, the ECRH extractor $\mathcal{E}_{j+1}$ is only guaranteed to output a preimage $\tilde{\ell}_{\mathbf{i}k}$ whenever the corresponding $\mathcal{E}_j$ outputs a vector in the range such that the ancestor $\tilde{\ell}_{\mathbf{i}}$ is a true image. For simplicity, we assume that in case $\mathcal{E}_j$ does not output a true image, $\mathcal{E}_{j+1}$ still outputs some string of length n^2 (without any guarantee on this string).

Overall, the witness extractor $\mathcal{E}_{\mathcal{P}^*}$ operates as follows. Given input $(1^n, e, C_{\mathbf{r}})$:

1. invoke $\mathcal{P}^*(e, C_{\mathbf{r}})$ and obtain (y, π) ;
2. obtain the depth $\tilde{d}$ from π , and abort if $\tilde{d} > 2cd$;
3. otherwise, extract the labels $(\tilde{\ell}_{\mathbf{i}k} : \mathbf{i} \in [n]^{\tilde{d}-1}, k \in [n]) \leftarrow \mathcal{E}_{\tilde{d}}(e, C_{\mathbf{r}})$;
4. letting $\tilde{\xi}$ be the extracted PCP-proof given by the leaves, apply the PCP witness extractor $\tilde{w} \leftarrow E_{\text{pcp}}(y, \tilde{\xi})$ and output $\tilde{w}$.

We now turn to prove that, except with negligible probability, whenever the verifier is convinced, the extractor $\mathcal{E}_{\mathcal{P}^*}$ outputs a valid witness. The proof is divided into two main claims as outlined in Section 5.1.3.

A reminder and some notation. Recall that prior to the prover's message, the randomness for the PCP verifier is of the form $\mathbf{r} = (r_i)_{i \in [q]} \in \{0, 1\}^{(\log^2 n) \times q}$ (and $q = \omega(\log n)$ is some fixed function). Once the verifier receives (y, π) , where $y = (\mathcal{M}, x, t)$ and $\pi = (\tilde{d}, \ell_{\emptyset}, C_{\text{dcom}})$, it computes the amount of coins required $\rho = O(\log t) < \log^2 n$ and interprets each r_j as its prefix $\hat{r}_j \in \{0, 1\}^{\rho}$. The corresponding PCP proof ξ (or the extracted $\tilde{\xi}$) is of size $n^{\tilde{d}+1}$. We shall denote by $\tilde{\xi} = \mathcal{E}_{\tilde{d}}(e, \text{PEnc}_R(\mathbf{r}))$ the full set of leaves extracted.

Relying on collision-resistance and ECRH extraction, we show that (almost) whenever the verifier is convinced, we extract a proof $\tilde{\xi}$ that locally satisfies the queries induced by the encrypted $\text{PEnc}_R(\mathbf{r})$.

Claim 5.3.2 (local consistency). *Let $\mathcal{P}^*$ be a polynomial-size prover strategy, where $|\mathcal{P}^*| \leq n^d$, and let $(\mathcal{E}_1, \dots, \mathcal{E}_{2cd})$ be its ECRH extractors as defined above. Then for all*

$n \in \mathbb{N}$,

$$\Pr_{(e,R,\mathbf{r}) \leftarrow \mathcal{G}(1^n)} \left[\begin{array}{l} (y, \pi) \leftarrow \mathcal{P}^*(e, \text{PEnc}_R(\mathbf{r})) \\ y = (\mathcal{M}, x, t), \pi = (\tilde{d}, \ell_\emptyset, C_{\text{dcom}}) \wedge \begin{array}{l} \tilde{\xi} \leftarrow \mathcal{E}_{\tilde{d}}(1^n, e, \text{PEnc}_R(\mathbf{r})) \\ V_{\text{pcp}}^{\tilde{\xi}}(y, \hat{\mathbf{r}}) = 0 \end{array} \\ \mathcal{V}_c(1^n, (e, R, \mathbf{r}), y, \pi) = 1 \end{array} \right] \leq \text{negl}(n) ,$$

where $\hat{\mathbf{r}} \in \{0, 1\}^{\rho \times q}$ consists of the prefixes of each $\mathbf{r} \in \{0, 1\}^{(\log^2 n) \times q}$.

Proof. Let us say that a tuple $(e, R, \mathbf{r})$ is “bad” if it leads to the above event. Assume towards contradiction that for infinitely many $n \in \mathbb{N}$, there is a noticeable fraction $\varepsilon(n)$ of bad tuples $(e, R, \mathbf{r})$. We show how to find collisions in $\mathcal{H}$.

Given e , sample coins R for the PIR encryption and coins $\mathbf{r}$ for the PCP verifier to simulate $\text{PEnc}_R(\mathbf{r})$. Given that the resulting $(e, R, \mathbf{r})$ is bad, let us show how to produce a collision in h_e .

First, invoke $\mathcal{P}^*(e, \text{PEnc}_R(\mathbf{r}))$ to obtain (y, π) , where $y = (\mathcal{M}, x, t)$ and $\pi = (\tilde{d}, \ell_\emptyset, C_{\text{dcom}})$. Next, decrypt C_{dcom} (using R) and obtain a set S of $q \cdot \text{polylog}(t)$ opened paths (each r_j in $\mathbf{r} = (r_i)_{i \in [q]}$ induces at most $\text{polylog}(t)$ queries). Each path corresponds to some leaf $\mathbf{i} \in [n]^{\tilde{d}}$ and contains $\tilde{d}$ strings of length n^2 each: $\mathbf{l}_1^{\mathbf{i}}, \dots, \mathbf{l}_{\tilde{d}}^{\mathbf{i}} \in \{0, 1\}^{n^2 \times \tilde{d}}$; each string $\mathbf{l}_j^{\mathbf{i}}$ contains the label of the j -th node along the path and the labels of all its siblings.

Next, note that $\tilde{d} \leq 2cd$. Indeed, if the verifier accepts then: $n^{\tilde{d}} \leq |x|^{2c}$, and in our case $|x| \leq |\mathcal{P}^*| \leq n^d$. Accordingly, we can invoke the extractors

$$\left\{ \mathcal{E}_j(e, \text{PEnc}_R(\mathbf{r}))[i_1, \dots, i_j] : j \in [\tilde{d}] \right\} ,$$

and for each opened path $\mathbf{i} \in S$, obtain the extracted $\tilde{\mathbf{l}}_1^{\mathbf{i}}, \dots, \tilde{\mathbf{l}}_{\tilde{d}}^{\mathbf{i}} \in \{0, 1\}^{n^2 \times \tilde{d}}$.

Let $\xi|_S = \left(\mathbf{l}_d^{\mathbf{i}} \right)_{\mathbf{i} \in S}$ be the leaves $\mathcal{P}^*$ opened and let $\tilde{\xi}|_S = \left(\tilde{\mathbf{l}}_d^{\mathbf{i}} \right)_{\mathbf{i} \in S}$ be the extracted leaves. Since $(e, R, \mathbf{r})$ are bad, it holds that $V_{\text{pcp}}^{\xi|_S}(x, \hat{\mathbf{r}}) = 1$ while $V_{\text{pcp}}^{\tilde{\xi}|_S}(x, \hat{\mathbf{r}}) = 0$; in particular, there exist some $\mathbf{i} \in S$ such that $\mathbf{l}_d^{\mathbf{i}} \neq \tilde{\mathbf{l}}_d^{\mathbf{i}}$. We focus from hereon on this specific $\mathbf{i}$. Let $j \in [\tilde{d}]$ be the smallest index such that $\mathbf{l}_j^{\mathbf{i}} \neq \tilde{\mathbf{l}}_j^{\mathbf{i}}$ (we just established that such an index exists); then it holds that $\mathbf{l}_{j-1}^{\mathbf{i}} = \tilde{\mathbf{l}}_{j-1}^{\mathbf{i}}$. Furthermore, since $(h, R, \mathbf{r})$ are bad, $\mathcal{V}$ accepts; this in turn implies that h_e compresses $\mathbf{l}_j^{\mathbf{i}}$ to the i_{j-1} -th block of $\mathbf{l}_{j-1}^{\mathbf{i}} = \tilde{\mathbf{l}}_{j-1}^{\mathbf{i}}$, which we will denote by ℓ^* . However, the latter is also the output of $\mathcal{E}_{j-1}(e, \text{PEnc}_R(\mathbf{r}))$, in coordinate $i_1 \dots i_{j-1}$, which in turn implies that $\mathcal{E}_j(e, \text{PEnc}_R(\mathbf{r}))[i_1 \dots i_{j-1}] = \tilde{\mathbf{l}}_j^{\mathbf{i}}$ is

also compressed by h_e to the same ℓ^* (except when extraction fails, which occurs with negligible probability). It follows that $\mathbf{I}_j^i \neq \tilde{\mathbf{I}}_j^i$ form a collision in h_e . $\square$

The second step in the proof of Proposition 106 is to show that, if the aforementioned extractor outputs a proof $\tilde{\xi}$ that convinces the PCP verifier with respect to the encrypted randomness, then the same proof $\tilde{\xi}$ must be globally satisfying (at least for witness extraction); otherwise, the polynomial-size extractor can be used to break the semantic security of the PIR.

Claim 5.3.3 (from local satisfaction to extraction). *Let $\mathcal{P}^*$ be a polysize prover and let $\mathcal{E}_{\mathcal{P}^*}$ be its polynomial-size extractor. Then for all $n \in \mathbb{N}$,*

$$\Pr_{(e, R, \mathbf{r}) \leftarrow \mathcal{G}(1^n)} \left[\begin{array}{ll} t \leq |x|^c & (y, \pi) \leftarrow \mathcal{P}^*(e, \text{PEnc}_R(\mathbf{r})) \\ V_{\text{pcp}}^{\tilde{\xi}}(y, \hat{\mathbf{r}}) = 1 & : y = (\mathcal{M}, x, t), \pi = (\tilde{d}, \ell_\emptyset, C_{\text{dcom}}) \\ E_{\text{pcp}}(y, \tilde{\xi}) \notin \mathcal{R}_c(y) & \tilde{\xi} \leftarrow \mathcal{E}_{\mathcal{P}^*}(1^n, e, \text{PEnc}_R(\mathbf{r})) \end{array} \right] \leq \text{negl}(n) ,$$

where $\hat{\mathbf{r}} \in \{0, 1\}^{\rho \times q}$ is the interpretation of $\mathbf{r} \in \{0, 1\}^{(\log^2 n) \times q}$ as a vector of prefixes (as detailed above).

Proof. Assume towards contradiction that for infinitely many $n \in \mathbb{N}$ the above event occurs with noticeable probability $\delta = \delta(n)$; we show how to break the semantic security of PEnc . First note that whenever the event occurs, by the PCP extraction guarantee, it holds that $\Pr_{\hat{\mathbf{r}} \leftarrow \{0, 1\}^{\rho \times q}} [V_{\text{pcp}}^{\tilde{\xi}}(y, \hat{\mathbf{r}}) = 1] \leq (1 - \varepsilon)^q$, where ε is the (constant) knowledge threshold of the PCP (see Section 5.3.2.3), and $q = \omega(\log n)$ is the number of repetitions.

To break semantic security, we consider the following CPA game, where a breaker $\mathcal{B}$ hands its challenger two independent strings of PCP randomness, $(\mathbf{r}_0, \mathbf{r}_1) \in \{0, 1\}^{(\log^2 n) \times 2}$, and gets back $\text{PEnc}_R(\mathbf{r}_b)$ for a random $b \in \{0, 1\}$. Now, $\mathcal{B}$ samples a random e , and runs $\mathcal{P}^*(e, \text{PEnc}_R(\mathbf{r}_b))$ and $\mathcal{E}_{\mathcal{P}^*}(1^n, e, \text{PEnc}_R(\mathbf{r}_b))$ to obtain an instance $y = (\mathcal{M}, x, t)$ from the prover and an extracted PCP proof $\tilde{\xi}$ from the extractor. Then, $\mathcal{B}$ computes the amount of coins required for V_{pcp} , $\rho = \rho(t)$, and derives the corresponding ρ -prefixes $(\hat{\mathbf{r}}_0, \hat{\mathbf{r}}_1)$ of $(\mathbf{r}_0, \mathbf{r}_1)$.

The breaker now runs the PCP extractor E_{pcp} on input $(y, \tilde{\xi})$ to obtain a string $\tilde{w}$ and verifies whether it is a valid witness for y (which can be done in $\text{poly}(|x|) = \text{poly}(n)$ time). In case the witness $\tilde{w}$ is valid or $V_{\text{pcp}}^{\tilde{\xi}}(y, \hat{\mathbf{r}}_0) = V_{\text{pcp}}^{\tilde{\xi}}(y, \hat{\mathbf{r}}_1)$, the breaker $\mathcal{B}$ outputs

a random guess for the bit b . Otherwise, the breaker outputs the single b' such that $V_{\text{pcp}}^{\tilde{\xi}}(y, \hat{\mathbf{r}}_{b'}) = 1$.

We now analyze the success probability of $\mathcal{B}$. We define two events F and E over a random choice of $(e, R, \hat{\mathbf{r}}_0, \hat{\mathbf{r}}_1, b)$; note that any choice of $(e, R, \hat{\mathbf{r}}_0, \hat{\mathbf{r}}_1, b)$ induces a choice of $y = (\mathcal{M}, x, t)$ and $\tilde{\xi}$. Define F to be the event that $t \leq |x|^c$ and $E_{\text{pcp}}(y, \tilde{\xi})$ fails to output a valid witness $\tilde{w}$; next, define E to be the event that $V_{\text{pcp}}^{\tilde{\xi}}(y, \hat{\mathbf{r}}_0) \neq V_{\text{pcp}}^{\tilde{\xi}}(y, \hat{\mathbf{r}}_1)$.

First, since we have assumed by way of contradiction that the event in the statement of the claim occurs with probability δ , we know that

$$\Pr [V_{\text{pcp}}^{\tilde{\xi}}(y, \hat{\mathbf{r}}_b) = 1 \mid F] = \frac{\delta}{\Pr[F]} .$$

Second, since E_{pcp} cannot extract a valid witness from $\tilde{\xi}$ and $\hat{\mathbf{r}}_{1-b}$ are random coins independent of $(\tilde{\xi}, y)$, we also know that

$$\Pr_{\hat{\mathbf{r}}_{1-b} \xleftarrow{U} \{0,1\}^{\rho \times q}} [V_{\text{pcp}}^{\tilde{\xi}}(y, \hat{\mathbf{r}}_{1-b}) = 1 \mid F] \leq (1 - \varepsilon)^q .$$

Combining these two facts, we deduce that

$$\begin{aligned} & \Pr [E \mid F] \\ & \geq \Pr [V_{\text{pcp}}^{\tilde{\xi}}(y, \hat{\mathbf{r}}_b) = 1 \wedge V_{\text{pcp}}^{\tilde{\xi}}(y, \hat{\mathbf{r}}_{1-b}) = 0 \mid F] \\ & \geq \Pr [V_{\text{pcp}}^{\tilde{\xi}}(y, \hat{\mathbf{r}}_b) = 1 \mid F] - \Pr [V_{\text{pcp}}^{\tilde{\xi}}(y, \hat{\mathbf{r}}_{1-b}) = 1 \mid F] \\ & \geq \Pr [V_{\text{pcp}}^{\tilde{\xi}}(y, \hat{\mathbf{r}}_b) = 1 \mid F] - (1 - \varepsilon)^q = \frac{\delta}{\Pr[F]} - \text{negl}(n) , \end{aligned}$$

so that, in particular, we can also deduce that

$$\Pr [F \wedge E] \geq \delta - \text{negl}(n) .$$

In addition, we deduce that

$$\begin{aligned}
& \Pr [\mathcal{B} \text{ guesses } b \mid F \wedge E] \\
& \geq 1 - \Pr [V_{\text{pcp}}^{\tilde{\xi}}(y, \hat{\mathbf{r}}_{1-b}) = 1 \mid F \wedge E] \\
& = 1 - \frac{\Pr [V_{\text{pcp}}^{\tilde{\xi}}(y, \hat{\mathbf{r}}_{1-b}) = 1 \wedge E \mid F]}{\Pr [E \mid F]} \\
& \geq 1 - \frac{(1 - \varepsilon)^q}{\delta / \Pr [F]} \\
& \geq 1 - \text{negl}(n) .
\end{aligned}$$

We now deduce that the breaker $\mathcal{B}$ guesses b with a noticeable advantage; indeed,

$$\begin{aligned}
\Pr [\mathcal{B} \text{ guesses } b] &= \Pr [F \wedge E] \Pr [\mathcal{B} \text{ guesses } b \mid F \wedge E] + \\
& \quad (1 - \Pr [F \wedge E]) \Pr [\mathcal{B} \text{ guesses } b \mid \bar{F} \vee \bar{E}] \\
&= \Pr [F \wedge E] \Pr [\mathcal{B} \text{ guesses } b \mid F \wedge E] + (1 - \Pr [F \wedge E]) \cdot \frac{1}{2} \\
&= \frac{1}{2} + \Pr [F \wedge E] \left(\Pr [\mathcal{B} \text{ guesses } b \mid F \wedge E] - \frac{1}{2} \right) \\
&\geq \frac{1}{2} + (\delta - \text{negl}(n)) \left(\frac{1}{2} - \text{negl}(n) \right) \\
&\geq \frac{1}{2} + \frac{\delta - \text{negl}(k)}{2} ,
\end{aligned}$$

thus completing the proof of the claim. $\square$

Putting it all together. By Claim 5.3.2 we conclude that whenever the verifier accepts, $\mathcal{E}_{\mathcal{P}^*}$ almost always extracts a proof $\tilde{\xi}$ which locally satisfies the PCP verifier on the encrypted randomness. By Claim 5.3.3, we deduce that whenever this occurs, $\tilde{\xi}$ must satisfy sufficiently many queries for PCP witness-extraction. This completes the proof of Proposition 106. $\square$

Universal efficiency. For input $y = (\mathcal{M}, x, t)$ (where $t < n^{\log n}$ for a security parameter n), the proof $\pi = (\ell_\emptyset, d, C_{\text{dcom}_\pi})$ is essentially dominated by the PIR answers C_{dcom_π} ; this includes $q = \text{polylog}(n)$ PIR answers for entries of size $\tilde{O}(n^2)$ (recall that $d = \log_n t < \log n$). In the PIR scheme of [48] the size of each PIR-answer is bounded by $E \cdot n \cdot \text{polylog}(n) + \log D$, where E is the size of an entry and D is the size of the entire DB. Hence, the overall length of the proof is bounded by a fixed polynomial $\tilde{O}(n^2)$,

independently of $|x|, |w|$ or c . The verifier's and prover's running time are bounded respectively by fixed universal polynomials $\text{poly}(|y|, n)$, $\text{poly}(n, t)$, again independently of c .

Remark 108 (Auxiliary input). *As noted in Remark 96, in our SNARK construction, we can actually settle for extraction with respect to restricted forms of auxiliary-input, for which no limitations are known. (The resulting SNARKs will account for the same auxiliary-input distributions.)*

Specifically, in our setting the extractor is required to handle an auxiliary input consisting of (honestly-generated) PIR-encryptions C of random strings, whose length is a-priori bounded by a fixed polynomial $\text{poly}(n)$ in the security parameter n . We can further use a PIR scheme where honestly generated ciphers are pseudo-random (e.g., the LWE-based PIR scheme of [48] can be setup to have this property), we can actually restrict the auxiliary input to be a random string, as long as the ECRH has efficient (even private) image verification. Indeed, an extractor that does noticeably better for random auxiliary-input than for pseudo-random auxiliary-input, can be used to distinguish the two cases. For this we should be able to efficiently check when that adversary produces an actual image (whereas the extractor fails to produce a corresponding preimage), which is why efficient image verification is needed.

Remark 109 (Parameter scaling). *In Kilian's original protocol, succinctness of the proof could be improved by making stronger hardness assumptions. For example, for security parameter n , if one is willing to assume collision-resistant hash functions with a $\text{polylog}(n)$ -long output, the proof length would be $\text{polylog}(n)$, rather than $\text{poly}(n)$. Unfortunately, in our construction we use a Merkle tree with $\text{poly}(n)$ fan-in; therefore, we cannot afford the same scaling. Specifically, even if we assume that our hash and PIR scheme have polylogarithmic-size output, each node in the Merkle tree still has $\text{poly}(n)$ siblings.⁸ Nonetheless, scaling can be performed if we make a stronger extractability assumption, such as the interactive one of [68], because in such a case there is no need to consider Merkle trees with polynomial fan-in as binary Merkle trees suffice for the security reduction.*

⁸We thank Kai-Min Chung and the anonymous referees of ITCS for pointing out the scaling problem.

5.3.3.3 Replacing ECRHs with GECRHs

We next explain why the *same construction* presented in the previous section also works when replacing ECRHs with generalized ECRHs (GECRHs) for any collision-resistant relation $\mathcal{R}^{\mathcal{H}}$.

First, observe that moving from ECRHs to GECRHs only affects the “local consistency” step of our proof (given by Claim 5.3.2). Indeed, in the proof based on ECRHs, the local-consistency step is where we employ collision-resistance to claim that the Merkle tree output by the extractor locally agrees with the opened paths (except with negligible probability).

The same argument still holds with GECRHs. Specifically, now given that the adversary produces an image $v = h_e(u)$, for $u \in \{0, 1\}^\ell$, the ECRH extractor outputs a “witness” $u' \in \{0, 1\}^{\ell+k}$, such that $\mathcal{R}_e^{\mathcal{H}}(v, u') = 1$. The SNARK witness extractor $\mathcal{E}_{\mathcal{P}^*}$, will reduce u' to its ℓ -bit prefix u'' , and proceed as before.

By the GECRH extraction guarantee, it must be that the hash image v of every node label u that appears in an opened path satisfies $\mathcal{R}_e^{\mathcal{H}}(v, u')$, where u' is the extracted node. By generalized collision-resistance, however, it must be that $u || 0^k = u'$; for if they were not, then we could utilize the prover and extractor for finding “generalized collisions”. Thus it follows that the prefix u'' of u' is the same as u , as in the previous proof. The other parts of the proof of remain unchanged.

Remark 110 (A weaker form of extraction). *For the SNARK application, and even weaker form of extraction (or generalized) extraction is sufficient. Indeed, in the described SNARK construction, the adversary implicitly outputs preimages of the hashes along the opened paths, under the PIR. More generally, we can thus relax the extraction requirement and ask that it only holds when the adversary outputs, not only an image, but also a preimage under some prespecified encoding function. This encoding function is typically hard to invert (e.g. a PIR encryption) and thus extraction does not become trivial. See more details in [29].*

5.4 Bootstrapping SNARKs

In this section, we show how any preprocessing SNARK can be *bootstrapped* into a fully succinct SNARK. As a result, relying on preprocessing SNARKs from the literature [34, 85, 111, 124], we obtain publicly-verifiable (fully succinct) SNARKs in the plain

model. More generally, we show that any SNARK can be bootstrapped into a new SNARK with *optimal* asymptotic efficiency, up to polynomial factors in the security parameter. In particular, the verifier runs in essentially linear time in its input, and the prover has to invest essentially as much time and space as in the original computation; thus we call these *complexity-preserving SNARKs*.

We first, describe in Section 5.4.1 our most basic transformation for publicly-verifiable preprocessing SNARKs; then in Section 5.4.2, we describe how to extend the transformation to the case of designated-verifier SNARKs; in Section 5.4.3, we address complexity-preserving SNARKs.

5.4.1 Bootstrapping Publicly-Verifiable Preprocessing SNARKs

In this section, we show

Theorem 111 (Bootstrapping publicly-verifiable SNARKs). *Any publicly-verifiable preprocessing SNARK for $\mathbf{NP}$ can be transformed into a publicly-verifiable fully-succinct SNARK for $\mathbf{NP}$.*

The theorem relies on two main parts. The first shows how to obtain publicly-verifiable *bounded-space* SNARKs *for read-once computations*. The second step is to show that SNARKs for such computations can be transformed to fully-succinct SNARKs.

5.4.1.1 SNARKs for Bounded-Space Read-Once Computations.

A bounded-space SNARK is a SNARK where the verifier and generator complexity depend on the space of the proven computation, but not on the time of the computation.

Definition 112. *A triple of algorithms $(\mathbb{G}, \mathbb{P}, \mathbb{V})$ is a **bounded-space SNARK** for $\mathbf{NP}$ if it is a SNARK where*

1. *The verifier $\mathbb{V}_c(\tau, \dots)$ only accepts instances $y = (\mathcal{M}, x, t)$ such that $|y| \leq 4n$, where n is the security parameter included in τ .*
2. *The following efficiency guarantee holds: there exists a universal polynomial p such that for any $c \in \mathbb{N}$, every large enough security parameter $n \in \mathbb{N}$, space bound $S \in \mathbb{N}$, and every instance $y = (\mathcal{M}, x, t) \in \mathcal{R}_c$, where $\mathcal{M}$ uses at most S space,*

- the generator $\mathbb{G}(1^n, S)$ runs in time $p(n + S)$;
- the prover $\mathbb{P}(\sigma, y, w)$ runs in time $p(n + t)$;
- the verifier $\mathbb{V}_c(\tau, y, \pi)$ runs in time $p(n + S)$;
- an honestly generated proof has size $p(n)$.

A read-once computation $\mathcal{M}(x, w)$ is a computation that can be divided into steps where at each step a single bit of the witness w is read, and never read again.

Definition 113 (Read-once computation). *A computation, given by a Turing Machine $\mathcal{M}$, input x , a time bound t , and a witness $w \in \{0, 1\}^t$, is said to be **read once** if it consists of t steps, where at step $j \in [t]$, a transition circuit $\widetilde{\mathcal{M}}_x(S_{j-1}, w_{j-1})$ takes as input a current state S_{j-1} and witness bit w_{j-1} , and outputs a next state S_j . The initial state is $S_0 = \text{init}$; the final state S_t is **acc** if $\mathcal{M}(x, w) = 1$, and **rej** otherwise. We further assume that if $|(\mathcal{M}, x, t)| \leq 4n$, and $\mathcal{M}$ uses at most S space, then $\widetilde{\mathcal{M}}_x$ is of size $\text{poly}(S, n)$, for some fixed universal polynomial poly .*

5.4.1.2 Constructing Bounded-Space SNARKs from Preprocessing SNARK, The Case of Public Verification

We show

Proposition 114. *Any publicly-verifiable preprocessing SNARK for **NP** can be transformed into a publicly-verifiable bounded-space SNARK for read-once computations in **NP**.*

We now describe the construction behind the proposition, following the high-level ideas presented in Section 5.1.4.

The computation tree. We consider a degree n tree structure of depth d constructed over an n^d -step computation. We label a node at level $j \in \{0, 1, \dots, d\}$ by $\mathbf{i} \in [n]^j$; for a leaf $\mathbf{i} \in [n]^d$, we denote by $\underline{\mathbf{i}}$ the number it represents in $\{0, 1, \dots, n^d - 1\}$; for any node $\mathbf{i} \in [n]^{d-j}$, denote by $\underline{\mathbf{i}}$ the number $\mathbf{i}'$ that its left most descendant leaf $\mathbf{i}' := (\mathbf{i}, 1^j)$ represents. Each node $\mathbf{i} \in [n]^{d-j}$ in the graph, at distance $d - j$ from the root, is associated with a pair of states $(S_{\underline{\mathbf{i}}}, S_{\underline{\mathbf{i}}+n^j})$, and represents the statement that $\mathcal{M}$, on input x , moves from $S_{\underline{\mathbf{i}}}$ to $S_{\underline{\mathbf{i}}+n^j}$, given some intermediate states and witness bits. Each leaf corresponds to two adjacent states in the computation and the root ε corresponds to the initial state and final state.

The prover $\mathbb{P}$ will compute a SNARK proof for each one of the statements along the tree. However, in order to avoid the full complexity of unwinding each of these computations, the proofs are computed recursively: at each intermediate node the prover proves that *it verified proofs attesting to the correctness of its descendents*.

We now describe the bounded-space SNARK $(\mathbb{G}, \mathbb{P}, \mathbb{V})$ in detail. The main technical component in the construction is the recursive-verification machine $\mathbb{M}$, defined next.

The recursive-verification machine $\mathbb{M}$. Let τ be a (public) verification state for a SNARK for $\mathbf{NP}$, and assume that a security parameter n is determined within τ . Let $y = (\mathcal{M}, x, t) \in \mathcal{L}_c$, where $\mathcal{M}$ is a read-once machine (Definition 113) with bounded space s , and $|y| \leq 4n$. The recursive-verification machine $\mathbb{M}^{y, \tau}$ takes input χ and witness ω . The input χ is a tuple $(\mathbf{i}, \mu, \mathbf{S}_{\mathbf{i}}, \mathbf{S}_{\mathbf{i}+n^j})$, where μ is a description of a machine, $t = n^d$, for some $d \in \mathbb{N}$,⁹ $\mathbf{i} \in [n]^{d-j}$, for some $j \in \{0, \dots, d\}$ and, $\mathbf{S}_{\mathbf{i}}, \mathbf{S}_{\mathbf{i}+n^j} \in \{0, 1\}^s$. The witness ω is a tuple $(\vec{\pi}, \vec{S}, w)$, where $\pi = (\pi_{\mathbf{i}+n^{j-1}}, \pi_{\mathbf{i}+2n^{j-1}}, \dots, \pi_{\mathbf{i}+n^j}) \in \{0, 1\}^{p(n) \times n}$ is a vector of n SNARK proofs, $\vec{S} = (\mathbf{S}_{\mathbf{i}+n^{j-1}}, \mathbf{S}_{\mathbf{i}+2n^{j-1}}, \dots, \mathbf{S}_{\mathbf{i}+(n-1)n^{j-1}}) \in \{0, 1\}^{s \times (n-1)}$ is a vector of $n-1$ intermediate states, and $w \in \{0, 1\}$ is a witness bit.

$\mathbb{M}^{y, \tau}$ recursively verifies the correctness of the computation given by y :

- **The halting condition:** for any leaf $\mathbf{i} \in [n]^d$, the machine verifies the correctness of the step specified by the leaf. That is, it verifies that $\widetilde{\mathcal{M}}_x(\mathbf{S}_{\mathbf{i}}, w) = \mathbf{S}_{\mathbf{i}+1}$.
- **The recursive step:** in any other node $\mathbf{i} \in [n]^{d-j}, j \in [d]$, the machine runs the SNARK verifier $\mathcal{V}_{\bar{c}}(\tau, \dots)$ to verify the proofs in $\vec{\pi}$ attesting to the correctness of the sub-computation corresponding to the subtree rooted at $\mathbf{i}$. Formally, the verifier $\mathcal{V}_{\bar{c}}$ verifies the computation given by the machine μ . (As we shall see when specifying $\mathbb{P}$ and $\mathbb{V}_c$, the recursion is carried out by letting μ be $\mathbb{M}^{y, \tau}$ itself.)

The formal description of $\mathbb{M}^{y, \tau}$ is given in Figure 5.2.

Parameter setting. We explain how to choose the constant $\bar{c}$ determining the relation $\mathcal{R}_{\bar{c}}$, and the time bound $t_{\mathcal{M}, \nu} = t_{\mathcal{M}, \nu}(n, S)$, referred to in Figure 5.2.

We would like $t_{\mathcal{M}, \nu}$ to bound the computation time of $\mathbb{M}^{y, \tau}(\chi, \omega)$, for any $\chi = (\mathbf{i}, \mu, \mathbf{S}, \mathbf{S}')$ and $\omega = (\vec{\pi}, \vec{S}, w)$ of interest. We would then like to choose $\bar{c}$ such that $t_{\mathcal{M}, \nu} \leq |\chi|^{\bar{c}}$, for any input χ of interest.

⁹The assumption that d has this form is *WLOG*, by adding dummy steps if needed.

Setting: Let $\mathcal{V}_{\bar{c}}$ be a SNARK verifier for $\mathcal{R}_{\bar{c}}$ (the choice of $\bar{c}$ is addressed in the *parameter setting* paragraph below). Let τ be a (public) verification state for $\mathcal{V}$, which includes the security parameter 1^n . Let $y = (\mathcal{M}, x, t)$, where $\mathcal{M}$ is a read-once machine, with bounded space $s \leq S \leq t \leq n^{\log n}$, and transition circuit $\widetilde{\mathcal{M}}_x$. Let d be such that $t = n^d$.

$\mathbb{M}^{y,\tau}(\chi, \omega)$:

1. **Parsing input and witness:** Parse χ as $(\mathbf{i}, \mu, S, S')$, and ω as $(\vec{\pi}, \vec{S}, w)$.
2. **Base case:** If $\mathbf{i} \in [n]^d$, verify that $\widetilde{\mathcal{M}}_x(S, w) = S'$.
3. **Recursive case:** If $\mathbf{i} \in [n]^{d-j}$, for some $j \in [d]$, then
 - Parse $\vec{\pi}$ as $(\pi_1, \dots, \pi_n)$, and $(S, \vec{S}, S')$ as $(S_0, S_1, \dots, S_n)$.
 - For each $k \in [n]$, verify that

$$\mathcal{V}_{\bar{c}}\left(\tau, (\mu, \chi_k, t_{\mathcal{M}, \mathcal{V}}), \pi_k\right) = 1 \quad ,$$

where $\chi_k = ((\mathbf{i}, k), \mu, S_{k-1}, S_k)$, and $t_{\mathcal{M}, \mathcal{V}}$ is a fixed polynomial in (n, S) , whose choice is addressed in the *parameter setting* paragraph.

Figure 5.2: The recursive-verification machine $\mathbb{M}^{y,\tau}$.

Recall that $y = (\mathcal{M}, x, t)$, and τ determines the security parameter n . The inputs and witness we will be interested in will satisfy $|\mu| \leq |\mathbb{M}^{y,\tau}|$, $|S| = |S'| \leq S \leq n^{\log n}$, $|\mathbf{i}| \leq |t| \leq \log^2 n$, and $|y| \leq 4n$.

We inspect the running time of $\mathbb{M}^{y,\tau}$ according to the two possible cases:

1. In the base case, the computation is dominated by running the transition circuit $\widetilde{\mathcal{M}}_x(S, w) = S'$. This computation is bounded by a fixed polynomial $\text{poly}_{\mathcal{M}}(n, S)$.
2. In the recursive case, verifies n SNARK proofs attesting to the correctness of the computation corresponding to a given subtree. The time required to do so depends on the running time of the SNARK verifier $\mathcal{V}_{\bar{c}}$, which can be bounded by a fixed polynomial $\text{poly}_{\mathcal{V}}(n, S)$.

Indeed, the running time of $\mathcal{V}_{\bar{c}}$ is bounded by a fixed polynomial $\text{poly}(|\chi|, |\mu|, n)$,

where $\chi = (\mathbf{i}, \mu, S, S')$, and

$$\begin{aligned} |S| &= |S'| \leq S, \\ |\mathbf{i}| &\leq \log^2 n, \\ |\mu| &\leq |\mathbb{M}^{y,\tau}| \leq \\ &O(|y| + |\tau| + |\widetilde{\mathcal{M}}_x| + \log t_{\mathcal{M},\mathcal{V}}) \leq \\ &n + \text{poly}(n) + \text{poly}_{\mathcal{M}}(n, S) + \log^2 n. \end{aligned}$$

Overall, there is a fixed polynomial $\text{poly}_{\mathcal{M},\mathcal{V}}$ so that we can set $t_{\mathcal{M},\mathcal{V}} = \text{poly}_{\mathcal{M},\mathcal{V}}(n, S)$.

It is left to show that we can choose $\bar{c}$ such that $t_{\mathcal{M},\mathcal{V}} \leq |\chi|^{\bar{c}}$ for any input of interest; indeed, by padding if necessary, we can ensure $|\chi| \leq (n + S)^{\Theta(1)}$, and thus we can choose $\bar{c}$ so that $t_{\mathcal{M},\mathcal{V}} \leq |\chi|^{\bar{c}}$. Crucially, the choice of the constant $\bar{c}$ does not affect the running time of $\mathcal{V}_{\bar{c}}$ (see Remark 91), and thus there is no circularity in this choice.

The bounded-space SNARK. We next describe in detail each components of the space-bounded SNARK $(\mathbb{G}, \mathbb{P}, \mathbb{V})$ for read once computations in **NP**, based on a publicly-verifiable preprocessing SNARK $(\mathcal{G}, \mathcal{P}, \mathcal{V})$ for $\mathcal{R}_{\bar{c}}$. The description is provided in Figure 5.3. As before, we shall consider a computation tree of arity n , and depth d where nodes are labeled by $\{\mathbf{i} \in [n]^j : 0 \leq j \leq d\}$. Also as above, for every leaf $\mathbf{i} \in [n]^d$, we denote by $\underline{\mathbf{i}}$ the corresponding number in $\{0, 1, \dots, n\}$, and for $\mathbf{i} \in [n]^{d-j}$, we denote by $\underline{\mathbf{i}}$ the number corresponding to its left most descendant leaf $(\mathbf{i}, 1^j)$.

Remark 115 (Prover complexity). *For simplicity of exposition, in the above description, we think about computing at each step, all the proofs corresponding to the current level of the tree. This may require that the prover uses space $t \gg S$. In Section 5.4.3, we explain how the tree can be computed in a more careful manner, using space $\tilde{O}(S)$.*

Proof of security. We now show that $(\mathbb{G}, \mathbb{P}, \mathbb{V})$ is a bounded-space SNARK for read-once computations in **NP**. The completeness and efficiency properties of the system immediately follow from those of the SNARK $(\mathcal{G}, \mathcal{P}, \mathcal{V})$. We concentrate on proving the adaptive proof of knowledge property.

Recall that for any $c \in \mathbb{N}$, and any (malicious) polysize prover $\mathbb{P}^*$, we would like to construct a corresponding polysize extractor $\mathbb{E}$ such that, when $\mathbb{P}^*$ convinces $\mathbb{V}_c$ of a statement $y = (\mathcal{M}, x, t)$, where $|y| \leq 4n$, $t \leq |x|^c$ and $\mathcal{M}$ uses space at most S , the

- **The generator.** Given the security parameter 1^n , and space bound S , the generator $\mathbb{G}$, computes $B = \text{poly}_{\mathcal{M}, \nu}(n, S)$, where $\text{poly}_{\mathcal{M}, \nu}$ is computed as described above, runs the preprocessing SNARK generator $\mathcal{G}(1^n, B)$ in order to obtain a reference string σ and (public) verification state τ , and then outputs (σ, τ) .

Without loss of generality, we shall assume that σ includes τ in the clear, and that τ includes the security parameter 1^n in the clear.

- **The prover.** Given $y = ((\mathcal{M}, x, t), w) \in \mathcal{R}_c$, where $|y| \leq 4n$, $w \in \{0, 1\}^t$ and $t = n^d$ for some $d \in \mathbb{N}$, the prover $\mathbb{P}$ gradually constructs the computation tree augmented with SNARK proofs attesting to the correctness of the computation given by y :

- For a leaf $\mathbf{i} \in [n]^d \setminus \{1^d\}$:
 1. Given state $S_{\mathbf{i}-1}$ of $\mathcal{M}(x, w)$, compute the next state $S_{\mathbf{i}} = \widetilde{\mathcal{M}}_x(S_{\mathbf{i}-1}, w_{\mathbf{i}})$.
 2. Invoke $\mathcal{P}(\sigma, \dots)$ to compute a SNARK proof $\pi_{\mathbf{i}}$ for $(y_{\mathbf{i}}, \omega_{\mathbf{i}}) \in \mathcal{R}_{\bar{c}}$, where $y_{\mathbf{i}} = (\mathbb{M}^{y, \tau}, \chi_{\mathbf{i}}, t_{\mathcal{M}, \nu})$, $\chi_{\mathbf{i}} = (\mathbf{i}, \mathbb{M}^{y, \tau}, S_{\mathbf{i}-1}, S_{\mathbf{i}}, t_{\mathcal{M}, \nu})$, $\omega_{\mathbf{i}} = (\perp, \perp, w_{\mathbf{i}})$, and $\bar{c}$, $t_{\mathcal{M}, \nu} = t_{\mathcal{M}, \nu}(n, S)$ were specified in the *parameter setting* paragraph.
- For a node $\mathbf{i} \in [n]^{d-j}$, where $j \in [d]$,
 1. Let $(S, \vec{S}, S') = (S_{\mathbf{i}}, S_{(\mathbf{i}, 1)}, \dots, S_{(\mathbf{i}, n-1)}, S_{(\mathbf{i}, n)})$, where $S_{(\mathbf{i}, k)} = S_{\mathbf{i} + kn^j}$.
 2. Let $\vec{\pi} = (\pi_{(\mathbf{i}, 1)}, \pi_{(\mathbf{i}, 2)}, \dots, \pi_{(\mathbf{i}, n)})$.
 3. Invoke $\mathcal{P}(\sigma, \dots)$ to compute a SNARK proof $\pi_{\mathbf{i}}$ for $(y_{\mathbf{i}}, \omega_{\mathbf{i}}) \in \mathcal{R}_{\bar{c}}$, where $y_{\mathbf{i}} = (\mathbb{M}^{y, \tau}, \chi_{\mathbf{i}}, t_{\mathcal{M}, \nu})$, $\chi_{\mathbf{i}} = (\mathbf{i}, \mathbb{M}^{y, \tau}, S, S', t_{\mathcal{M}, \nu})$, $\omega_{\mathbf{i}} = (\vec{\pi}, \vec{S}, \perp)$.
- The final proof is π_{ε} corresponding to the root ε .

- **The verifier.** Given c , $y = (\mathcal{M}, x, t)$, and proof π_{ε} ,

1. Verify that $|y| \leq 4n$, $t \leq |x|^c$, and that $\mathcal{M}$ uses space at most $S \leq t$.^a
2. Verify that

$$\mathcal{V}_{\bar{c}}\left(\tau, (\mathbb{M}^{y, \tau}, \chi_{\varepsilon}, t_{\mathcal{M}, \nu}), \pi_{\varepsilon}\right) = 1 \quad ,$$

where $\chi_{\varepsilon} = (\varepsilon, \mathbb{M}^{y, \tau}, \text{init}, \text{acc})$, init and acc represent the initial state and final accepting state of $\mathcal{M}$ (respectively), and $t_{\mathcal{M}, \nu} = t_{\mathcal{M}, \nu}(n, S)$ were specified in the *parameter setting* paragraph.

^aWe insist that there is an explicit space bound in $\mathcal{M}$'s description.

Figure 5.3: A bounded-space SNARK for read once computations in **NP**

extractor can find a witness $w \in \{0, 1\}^t$, such that $(y, w) \in \mathcal{R}_c$, with all but negligible probability. To achieve this goal, we employ a recursive extraction strategy, which we now describe.

Before describing the extraction procedure, we state a useful lemma concerning multiple-witness extraction (similar to Lemma 107). Specifically, our extraction procedure will be based on recursive extraction down the computation tree; we will be interested in provers producing a vector of instances $\vec{y}$ and SNARK proofs $\vec{\pi}$. Here it will be convenient to invoke an extractor that can extract a vector of witnesses $\vec{w}$ con-

taining a witness for each accepting statement in $\vec{y}$. This notion of extraction follows rather directly from the “single-instance” extraction notion of Definition 89.

Lemma 116 (adaptive proof of knowledge for instance vectors). *Let $(\mathcal{G}, \mathcal{P}, \mathcal{V})$ be a SNARK for **NP** (as in Definition 89). Then for any $c \in \mathbb{N}$, and any polynomial-size prover $\mathcal{P}^*$ there exists a polynomial-size extractor $\mathbb{E}_{\mathcal{P}^*}$ such that for any security parameter $n \in \mathbb{N}$, every auxiliary input $z \in \{0, 1\}^{\text{poly}(n)}$, and every time bound $B \in \mathbb{N}$,*

$$\Pr \left[\begin{array}{c} \exists i \text{ s.t.} \\ \mathcal{V}_c(\tau, y_i, \pi_i) = 1 \\ (y_i, w_i) \notin \mathcal{R}_c \end{array} \middle| \begin{array}{l} (\sigma, \tau) \leftarrow \mathcal{G}(1^n, B) \\ (\vec{y}, \vec{\pi}) \leftarrow \mathcal{P}^*(z, \sigma) \\ \vec{w} \leftarrow \mathcal{E}_{\mathcal{P}^*}(z, \sigma) \end{array} \right] \leq \text{negl}(n) .$$

We now turn to describe the extractor $\mathbb{E}_{\mathcal{P}^*}$. Given the prover $\mathcal{P}^*$, we construct c (families of) polysize extractors $\mathcal{E}_1, \dots, \mathcal{E}_c$, one for each potential level in the computation tree. To make notation lighter, we do not explicitly write the auxiliary input z that may be given to $\mathcal{P}^*$ and its extractor $\mathbb{E}_{\mathcal{P}^*}$ (e.g., any random coins used by $\mathcal{P}^*$); similarly for the SNARK provers and their extractors discussed below. The proof naturally holds for any auxiliary input distribution $\mathcal{Z}$, provided that the underlying SNARK is secure with respect to $\mathcal{Z}$.

The extractors are constructed as follows:

- Use the prover $\mathcal{P}^*$ to construct a SNARK prover $\mathcal{P}_1^*$ that works as follows: on input σ , $\mathcal{P}_1^*$ computes $(y, \pi_\varepsilon) \leftarrow \mathcal{P}^*(\sigma)$, constructs the instance $y_\varepsilon := (\mathbb{M}^{y, \tau}, (\varepsilon, \mathbb{M}^{y, \tau}, \text{init}, \text{acc}), t_{\mathcal{M}, \mathcal{V}})$, and outputs $(y_\varepsilon, \pi_\varepsilon)$.

Then define $\mathcal{E}_1 := \mathcal{E}_{\mathcal{P}_1^*}$ to be the SNARK extractor for the SNARK prover $\mathcal{P}_1^*$. Like $\mathcal{P}_1^*$, $\mathcal{E}_1$ also expects input σ ; $\mathcal{E}_1$ returns $(\vec{\pi}, \vec{S}, w)$ that is (with all but negligible probability) a valid witness for the SNARK statement y_ε , assuming that $\mathbb{V}_{\bar{c}}$ (and thus also $\mathcal{V}_{\bar{c}}$) accepts π_ε .

- Use $\mathcal{E}_1$ to construct the new SNARK prover $\mathcal{P}_2^*$ that works as follows: on input σ , $\mathcal{P}_2^*$ computes $(\vec{\pi}, \vec{S}, w) \leftarrow \mathcal{E}_1(\sigma)$ and then outputs $(\vec{y}, \vec{\pi})$, where the vector of SNARK statements $\vec{y}$ contains an entry $y_i := (\mathbb{M}^{y, \tau}, (i, \mathbb{M}^{y, \tau}, \mathbf{S}_{i-1}, \mathbf{S}_i), t_{\mathcal{M}, \mathcal{V}})$ for each $i \in [n]$. Then define $\mathcal{E}_2 := \mathcal{E}_{\mathcal{P}_2^*}$ to be the SNARK extractor for the SNARK prover $\mathcal{P}_2^*$. Given σ (with all but negligible probability) $\mathcal{E}_2$ should output a witness for each accepting statement and convincing proof (y_i, π_i) in $(\vec{y}, \vec{\pi})$.

- In general, for each $1 < j \leq c$, we similarly define $\mathcal{P}_j^*$ that computes $\{(\vec{\pi}^{(i)}, \vec{S}^{(i)}, w^{(i)}) : i \in [n]^{j-1}\} \leftarrow \mathcal{E}_{j-1}(\sigma)$ and then outputs $\{(\vec{y}^{(i)}, \vec{\pi}^{(i)}) : i \in [n]^{j-1}\}$, where the vector of SNARK statements $\vec{y}^{(i)}$ contains an entry $y_{(i,k)} := (\mathbb{M}^{y,\tau}, ((i,k), \mathbb{M}^{y,\tau}, S_{k-1}^{(i)}, S_k^{(i)}), t_{\mathcal{M},\nu})$ for each $k \in [n]$. Then define $\mathcal{E}_j := \mathcal{E}_{\mathcal{P}_j^*}$ to be the SNARK extractor for the SNARK prover $\mathcal{P}_j^*$.

We now define the extractor $\mathbb{E}_{\mathbb{P}^*}$. On input σ , $\mathbb{E}_{\mathbb{P}^*}$ runs $\mathbb{P}^*(\sigma)$ and obtains the instance $y = (\mathcal{M}, x, t)$. It then computes $d := \log_n t$. If $d > c$ $\mathbb{E}_{\mathbb{P}^*}$ aborts, and otherwise it runs $\mathcal{E}_d(\sigma)$ and obtains $\{(\vec{\pi}^{(i)}, \vec{S}^{(i)}, w^{(i)}) : i \in [n]^d\}$. It then outputs as the witness $w = \{w^{(i)} : i \in [n]^d\}$ as the witness for y .

Note that each $\mathcal{E}_j$ is of polynomial size, because each $\mathcal{E}_j$ is constructed via a constant number of recursive invocations of the SNARK proof of knowledge, and each such invocation incurs an arbitrary polynomial blowup in the size of the extractor relative to its prover. Thus, we deduce that $\mathbb{E}_{\mathbb{P}^*}$ is also of polynomial size.

It is left to argue that w extracted by $\mathbb{E}_{\mathbb{P}^*}$ is a valid witness for y :

Proposition 117. *Let $\mathbb{P}^*$ be a polysize prover for the bounded-space SNARK, and let $\mathbb{E}_{\mathbb{P}^*}$ be its corresponding extractor as defined above. Then for any $n \in \mathbb{N}$, and space bound $S \in \mathbb{N}$:*

$$\Pr \left[\begin{array}{c} \mathbb{V}_c(\tau, y, \pi) = 1 \\ (y, w) \notin \mathcal{R}_c \end{array} \middle| \begin{array}{c} (\sigma, \tau) \leftarrow \mathbb{G}(1^n, S) \\ (y, \pi) \leftarrow \mathbb{P}^*(\sigma) \\ w \leftarrow \mathbb{E}_{\mathbb{P}^*}(\sigma) \end{array} \right] \leq \text{negl}(n) .$$

Proof. We assume throughout that $|y| \leq 4n$, $t = n^d$, for $d \leq c$, and that $\mathcal{M}$ is a read-once machine that uses space at most S ; indeed, $\mathbb{V}$ explicitly checks all of these conditions. For each node $i \in [n]^j, j \in \{0, \dots, d\}$, we consider the corresponding input $\chi_i := (\mathbb{M}^{y,\tau}, (i, \mathbb{M}^{y,\tau}, S_i, S_{i+n^{d-|i|}}), t_{\mathcal{M},\nu})$ and witness $\omega_i = (\vec{\pi}^{(i)}, \vec{S}^{(i)}, w^{(i)})$ induced by the recursive extraction procedure, and show that $\mathbb{M}^{y,\tau}(\chi_i, \omega_i) = 1$. If this is the case, then by the construction of $\mathbb{M}$, the extracted $\{S_i, S_{i+1}, w^{(i)} : i \in [n]^d\}$, satisfy

$$S_0 = \text{init}, S_i = \widetilde{\mathcal{M}}_x(S_{i-1}, w^{(i)}), S_t = \text{acc} ,$$

Implying that $\mathcal{M}(y, w) = 1$.

The fact that $\mathbb{M}^{y,\tau}(\chi_i, \omega_i) = 1$ is shown by induction on the level of the computation

tree. For the base case, we show that, except with negligible probability, whenever the prover $\mathbb{P}^*$ convinces the verifier $\mathbb{V}$ to accept (y, π) , the extractor $\mathcal{E}_1$ outputs $\omega_\varepsilon = (\vec{\pi}, \vec{S}, w)$ such that, for each $i \in [n]$, and letting $y_i := (\mathbb{M}^{y,\tau}, (i, \mathbb{M}^{y,\tau}, S_{i-1}, S_i), t_{\mathcal{M},\nu})$, it holds that $\mathcal{V}_{\bar{c}}(\tau, y_i, \pi_i) = 1$.

Indeed, the fact that $\mathbb{V}_c(\tau, y, \pi) = 1$ implies that $\mathcal{V}_{\bar{c}}(\tau, y_\varepsilon, \pi) = 1$, where $y_\varepsilon = (\mathbb{M}^{y,\tau}, (\varepsilon, \mathbb{M}^{y,\tau}, \text{init}, \text{acc}), t_{\mathcal{M},\nu})$. By the SNARK proof of knowledge property, whenever $\mathbb{V}$ accepts, with all but negligible probability, the extractor $\mathcal{E}_1$ outputs a valid witness $\omega_\varepsilon = (\vec{\pi}, \vec{S}, w)$ for the statement y_ε , implying that $\mathbb{M}^{y,\tau}(\chi_\varepsilon, \omega_\varepsilon) = 1$. By construction of the machine $\mathbb{M}^{y,\tau}$, the extracted witness satisfies $\mathcal{V}_{\bar{c}}(\tau, y_i, \pi_i) = 1$, for any $i \in [n]$, as claimed.

For the inductive step, assume that $\mathbb{M}^{y,\tau}(\chi_{\mathbf{i}}, \omega_{\mathbf{i}}) = 1$, for $\mathbf{i} \in [n]^j, j \in [d]$, input $\chi_{\mathbf{i}} := (\mathbb{M}^{y,\tau}, (\mathbf{i}, \mathbb{M}^{y,\tau}, S_{\mathbf{i}}, S_{\mathbf{i}+n^{d-j}}), t_{\mathcal{M},\nu})$ and witness $\omega_{\mathbf{i}} = (\vec{\pi}^{(\mathbf{i})}, \vec{S}^{(\mathbf{i})}, w^{(\mathbf{i})})$. Then, by the definition of $\mathbb{M}^{y,\tau}$, for any $k \in [n]$, $y_{(\mathbf{i},k)} := (\mathbb{M}^{y,\tau}, ((\mathbf{i}, k), \mathbb{M}^{y,\tau}, S_{k-1}^{(\mathbf{i})}, S_k^{(\mathbf{i})}), t_{\mathcal{M},\nu})$, it holds that $\mathcal{V}_{\bar{c}}(\tau, y_{(\mathbf{i},k)}, \pi_k^{(\mathbf{i})}) = 1$. Then by the SNARK proof of knowledge guarantee, the extractor $\mathcal{E}_{j+1}$ outputs a witness $\omega_{(\mathbf{i},k)}$, for each $\chi_{(\mathbf{i},k)} = ((\mathbf{i}, k), \mathbb{M}^{y,\tau}, S_{k-1}^{(\mathbf{i})}, S_k^{(\mathbf{i})})$, such that $\mathbb{M}^{y,\tau}(\chi_{(\mathbf{i},k)}, \omega_{(\mathbf{i},k)}) = 1$, as required.

This concludes the proof of Proposition 117. $\square$

5.4.1.3 From Bounded-Space SNARKs to Fully-Succinct SNARKs

In this section, we show that bounded-space SNARKs as constructed in the previous subsection are sufficient for obtaining fully-succinct SNARKs.

Proposition 118. *Assuming the existence of collision-resistant hashing, any bounded-space SNARK for read-once computations in $\mathbf{NP}$ can be converted into a fully-succinct SNARK for $\mathbf{NP}$.*

The transformation behind the proposition consists of two steps. The first step is to reduce any instance $y = (\mathcal{M}, x, t)$, which may such that $|y| \gg n$, into an instance $y' = (\mathcal{M}', x', t')$ of size, say, $3n$.

The second step is to reduce any instance $y = (\mathcal{M}, x, t)$, where $\mathcal{M}$ might potentially use a large amount of space, into a new instance $y' = (\mathcal{M}', x', t')$, where $\mathcal{M}'$ only uses bounded space and runs in similar time up to factors in the security parameter.

Both reductions are *computational* in the sense that any computationally bounded adversary that produces an accepting witness w' for y' can be used to produce a witness w for y , assuming it cannot break some underlying computational assumption.

The first transformation - input checking machines. The transformation operates by taking a potentially large instance $y = (\mathcal{M}, x, t)$, and compressing it into $v = h_e(y)$ using a collision-resistant hash function. Then we consider a new machine $\mathcal{M}'_{e,v}(1^n, w')$ that expects a witness $w' = (y, w)$, and first verifies that $h_e(y) = v$.

Lemma 119. *Let $\mathcal{H}$ be a family of collision-resistant hash functions. There exist efficiently computable (deterministic) functions M'_1, W'_1, W_1 and T'_1 such that:*

1. Syntax: *For every Turing machine $\mathcal{M}$, $n \in \mathbb{N}$, $e \in \text{supp}(\mathcal{K}_{\mathcal{H}}(1^n))$, $\mathcal{M}' := M'_1(y, e)$ is a turing machine.*

2. Witness Reductions:

- *For every $n \in \mathbb{N}$, instance $y = (\mathcal{M}, x, t) \in \mathcal{R}_{\mathcal{U}}$, witness $w \in \mathcal{R}_{\mathcal{U}}(y)$, $e \in \text{supp}(\mathcal{K}_{\mathcal{H}}(1^n))$, let $w' = W'_1(y, w)$, and $t' = T'_1(t, |y|, n)$. Then,*

$$((\mathcal{M}', 1^n, t'), w') \in \mathcal{R}_{\mathcal{U}} .$$

- *For every $c \in \mathbb{N}$, every polynomial-size adversary $\mathcal{A}$, and $n \in \mathbb{N}$,*

$$\Pr_{e \leftarrow \mathcal{K}_{\mathcal{H}}(1^n)} \left[\begin{array}{c} (y', w') \in \mathcal{R}_c \\ (y, w) \notin \mathcal{R}_{\mathcal{U}} \end{array} \middle| \begin{array}{c} (y, w') \leftarrow \mathcal{A}(e) \\ y = (\mathcal{M}, x, t) \\ y' = (M'_1(y, e), x, T'_1(t, |y|, n)) \\ w \leftarrow W_1(w') \end{array} \right] \leq \text{negl}(n) .$$

3. Efficiency:

- $|y'| = |\mathcal{M}'| + n + |t'| = 3n$.
- $T'_1(t, |y|, n) = t + |y| \cdot n^{c_{\mathcal{H}}}$, where $c_{\mathcal{H}}$ is a constant that depends only on $\mathcal{H}$;
- The transformations $M'_1(y, e)$ runs in linear time;
- The transformations $W'_1(y, w)$, $W_1(w')$ run in linear time and logarithmic space.

Proof sketch. Assume that for security parameter n , each $e \in \text{supp}(\mathcal{K}_{\mathcal{H}}(1^n))$ is of length n and $h_e : \{0, 1\}^* \rightarrow \{0, 1\}^n$. The machine $\mathcal{M}' = M'_1(y, e)$ will have $v = h_e(y)$ hardwired into its description, given input $(1^n, w')$, it first parses $w' = (y, w)$, checks

that $h_e(y) = v$, and then emulate $\mathcal{M}(y, w)$. We then naturally define $W'_1(y, w) = (y, w)$, and $W_1(y, w) = w$. $T'_1(t, y, n) = t + |y| \cdot \text{poly}(n)$, where poly is a fixed polynomial determined by the hashing family $\mathcal{H}$.

The witness reduction and efficiency properties readily follow from the collision-resistance and efficiency of $\mathcal{H}$. $\square$

The second transformation - machines with unreliable memory. We rely on the *online memory checking* techniques of Blum et al. [39], which show how computations relying on *large* reliable memory can be converted into computations that only rely on a *small* reliable memory, and large unreliable memory that may be tampered with. Under computational assumptions, and assuming that tampering is bounded, this transformation only introduces polynomial overhead in the security parameter, independently of the time or space used in the computation.

Ben-Sasson et al. [24] formalize the above for non-deterministic random-access machines, under the assumption of collision-resistant hashing. As in the rest of this thesis, here we shall restrict attention to Turing machines.

Lemma 120 (Follows from [39],[24]). *Let $\mathcal{H}$ be a family of collision-resistant hash functions. There exist efficiently computable (deterministic) functions M'_2, W'_2, W_2 and T'_2 such that:*

1. Syntax: *For every Turing machine $\mathcal{M}$, $n \in \mathbb{N}$, $e \in \text{supp}(\mathcal{K}_{\mathcal{H}}(1^n))$, $\mathcal{M}' := M'_2(\mathcal{M}, e)$ is a read-once turing machine.*

2. Witness Reductions:

- *For every $n \in \mathbb{N}$, instance $y = (\mathcal{M}, x, t) \in \mathcal{R}_{\mathcal{U}}$, witness $w \in \mathcal{R}_{\mathcal{U}}(y)$, $e \in \text{supp}(\mathcal{K}_{\mathcal{H}}(1^n))$, let $w' = W'_2(y, w, e)$, and $t' = T'_2(t, n)$. Then,*

$$((\mathcal{M}', x, t'), w') \in \mathcal{R}_{\mathcal{U}} .$$

- For every $c \in \mathbb{N}$, every polynomial-size adversary $\mathcal{A}$, and $n \in \mathbb{N}$,

$$\Pr_{e \leftarrow \mathcal{K}_{\mathcal{H}}(1^n)} \left[\begin{array}{c|c} (y', w') \in \mathcal{R}_c & (y, w') \leftarrow \mathcal{A}(e) \\ (y, w) \notin \mathcal{R}_{\mathcal{U}} & y = (\mathcal{M}, x, t) \\ & y' = (M'_2(\mathcal{M}, e), x, T'_2(t, n)) \\ & w \leftarrow W_2(y, w', e) \end{array} \right] \leq \text{negl}(n) .$$

3. Efficiency:

- $\mathcal{M}' = M'_2(\mathcal{M}, e)$ is a $\text{poly}(n)$ -space machine and $T'_2(t, n) = t \cdot n^{c_{\mathcal{H}}}$, where $c_{\mathcal{H}}$ is a constant that depends only on $\mathcal{H}$; also $|y'| = |\mathcal{M}'| + |x| + |t| \leq |y| + n$.
- The transformations $M'_2(\mathcal{M}, e)$ and $W(y, w', e)$ run in linear time;
- The transformation $W'_2(y, w, e)$ runs in time $(|\mathcal{M}| + |x| + t) \cdot \text{poly}(n)$ and space $(|\mathcal{M}| + |x| + s) \cdot \text{poly}(n)$, where s is the space complexity of $\mathcal{M}(x, w)$.

The high-level idea. The idea behind the lemma is to transform $\mathcal{M}$ into a machine $\mathcal{M}'$ that uses the hash function h_e to delegate its memory into the “untrusted witness” w' by dynamically maintaining a Merkle tree over w' . Every time the machine reads a bit from its memory in the untrusted witness, it first authenticates the bit according to the currently held Merkle root; the authentication information itself is provided within w' . Likewise, when it writes a bit, it updates the Merkle root. These operation only requires locally handling hash values along any given path, and can be done with fixed $\text{poly}(n)$ space.

We can further assume WLOG that $\mathcal{M}'$ is a read-once machine, by first augmenting $\mathcal{M}$ so that every witness bit that it reads, is written into its memory.

For more details details see [24, 39].

Putting the transformations together. We now describe in Figure 5.4 how given a bounded-space SNARK $(\mathbb{G}, \mathbb{P}, \mathbb{V})$ for read once computations in **NP**, we invoke the above transformations to obtain a fully-succinct SNARK $(\mathcal{G}, \mathcal{P}, \mathcal{V})$ for **NP**.

Parameters and completeness. The function $c(|y|)$ essentially reflects the overhead in computation time given by the two transformations. For completeness, we need to choose it so ensure that whenever $y \in \mathcal{L}_c$, then $y_2 \in \mathcal{L}_{c \cdot c(|y|)}$; indeed, choosing

- **The generator.** Given the security parameter 1^n , let $S(n) = \text{poly}(n)$ be the fixed polynomial bound on the space given by transformation from Lemma 120. The generator $\mathcal{G}$ invokes the bonded-space SNARK generator $\mathcal{G}(1^n, S(n))$ in order to obtain a reference string σ and verification state τ . In addition, it samples a key for a collision-resistant hash $e \leftarrow \mathcal{K}_{\mathcal{H}}(1^n)$. It outputs (σ, e) as the reference string and (τ, e) as the verification state.
- **The prover.** Given $y = ((\mathcal{M}, x, t), w) \in \mathcal{R}_c$,
 1. Apply first transformation to (y, w) to obtain

$$(y_1, w_1) = ((\mathcal{M}_1, x_1, t_1), w_1) = ((M'_1(y, e), 1^n, T'_1(t, |y|, n)), W'_1(y, w)) \ .$$
 2. Apply second transformation to (y_1, w_1) to obtain

$$(y_2, w_2) = ((\mathcal{M}_2, x_2, t_2), w_2) = ((M'_2(\mathcal{M}_1, e), x_1, T'_2(t, n)), W'_2(y, w, e)) \ .$$
 3. Invoke $\mathbb{P}(\sigma, y_2, w_2)$ to compute a SNARK proof π attesting that $y_2 \in \mathcal{R}_{\mathcal{U}}$.
- **The verifier.** Given c , $y = (\mathcal{M}, x, t)$, and proof π ,
 1. Verify that $t \leq |x|^c$.
 2. Apply both transformations to y to obtain y_2 .
 3. Invoke $\mathbb{V}_{c \cdot c(|y|)}(\tau, y_2, \pi)$, for $c(|y|) = c_{\mathcal{H}} \cdot \log_n |y|$, such that $c_{\mathcal{H}}$ is the fixed constant depending only on the hashing family $\mathcal{H}$ addressed in Lemmas 119, 120.

Figure 5.4: A fully-succinct SNARK for **NP**

$c(|y|) = c_{\mathcal{H}} \cdot \log_n |y|$, by the efficiency properties of two transformations, we have:

$$\begin{aligned}
 t_1 &\leq \\
 t + |y| \cdot n^{c_{\mathcal{H}}} &\leq \\
 |x|^c + n^{c(|y|) \cdot c_{\mathcal{H}}^{-1} + c_{\mathcal{H}}} &\leq \\
 n^{c(|y|) \cdot c_{\mathcal{H}}^{-1} \cdot c} + n^{c(|y|) \cdot c_{\mathcal{H}}^{-1} + c_{\mathcal{H}}} & \\
 t_2 &\leq \\
 t_1 \cdot n^{c_{\mathcal{H}}} &\leq \\
 n^{c(|y|) \cdot c} &\ .
 \end{aligned}$$

Since $y_2 = (\mathcal{M}_2, 1^n, t_2)$, it follows that $y_2 \in \mathcal{L}_{c \cdot c(|y|)}$, as required.

Also for completeness, we need make sure that $\mathcal{M}_2$ is a read-once machine that uses space at most $S(n)$, which is, indeed, guaranteed by the second transformation. We

also need to make sure that $|y_2| \leq 4n$. Indeed

$$\begin{aligned} |y_1| &= 3n \\ |y_2| &\leq |y_1| + n \leq 4n . \end{aligned}$$

Proof of knowledge. We now sketch why the proof of knowledge property holds. Note that for any polysize $\mathcal{P}^*$, there exists a constant $d \in \mathbb{N}$, such that for any security parameter n , the corresponding circuit $\mathcal{P}_n^*$ is of size at most n^d . In particular, the size of any y it outputs is bounded by n^d , and $c' = c \cdot c(|y|)$ is bounded by $c \cdot c_{\mathcal{H}} \cdot d$. Thus, we can think of $\mathcal{P}^*$ as a malicious prover $\mathbb{P}^*$ for the bounded-space SNARK, and can invoke the extractor $\mathbb{E}_{\mathbb{P}^*}$, with respect to the constat c' and the relation $\mathcal{R}_{c'}$.

By the extraction guarantee, whenever $\mathcal{V}_c$ and thus $\mathbb{V}_{c'}$ is convinced, $\mathbb{E}_{\mathbb{P}^*}$ will produce a witness w_2 for y_2 . We then apply the two efficient transformations, first W_2 , and the W_1 , to obtain a witness w for the original instance y . Since $\mathbb{E}_{\mathbb{P}^*}$ is of polysize, we are guaranteed that, with all but negligible probability, indeed $w \in \mathcal{R}_c(y)$.

5.4.2 Bootstrapping Designated-Verifier Preprocessing SNARKs

In the previous section, we have proved a bootstrapping theorem for publicly-verifiable SNARKs. In this section, we show a bootstrapping theorem for the case of designated-verifier SNARKs.

Theorem 121 (Bootstrapping designated-verifier SNARKs). *Any designated-verifier preprocessing SNARK for $\mathbf{NP}$ can be transformed into a designated-verifier fully-succinct SNARK for $\mathbf{NP}$.*

Like in the case of public verification, the theorem relies on two parts. The first shows how to obtain designated-verifier *bounded-space* SNARKs *for read-once computations*. The second step is to show that SNARKs for such computations can be transformed to fully-succinct SNARKs. The second step is, in fact, identical in both cases of public verification and designated verification. We thus focus on the first step.

5.4.2.1 Constructing Bounded-Space SNARKs from Preprocessing SNARK, The Case of Designated Verification

We show

Proposition 122. *Any designated-verifier preprocessing SNARK for $\mathbf{NP}$ can be transformed into a designated-verifier bounded-space SNARK for read-once computations in $\mathbf{NP}$.*

The challenge and its answer. When we try to adapt the bounded-space SNARK construction for the publicly-verifiable case (see Section 5.4.1.2) to the designated-verifier case, we encounter the following difficulty: how can we prove in an intermediate node in the computation tree that we have verified a proof for the correctness of the computation in its corresponding subtree, when the prover cannot verify the proof on its own? More concretely: the prover does not know the verification state τ (because it is secret) and, therefore, cannot provide a witness for such a theorem.

We show how to circumvent this difficulty, using fully-homomorphic bit encryption (FHE). The high-level idea is as follows. We encrypt the private verification state τ and attach its encryption c^τ to the public reference string σ . Then, in any intermediate node, instead of verifying the proofs received from its sons, we homomorphically evaluate the SNARK verifier $\mathcal{V}$ on the encrypted verification state c^τ , as well as the statement and proof at hand. In order to enforce correctness, the prover provides at each node, as part of the proof, the result of the homomorphic evaluation $\hat{c}$, and a proof that it “knows” a previous proof, such that $\hat{c}$ is indeed the result of evaluating $\mathcal{V}$ on c^τ on this proof (and some corresponding statement). The verifier $\mathbb{V}$, can apply the SNARK verifier $\mathcal{V}$ to check that: (a) the SNARK proof is valid, (b) the decryption of $\hat{c}$ is indeed “1”. Doing the above as is, may cause the size of the proofs to blow up due to appending $\hat{c}$ at each node. We avoid this by homomorphically multiplying the encryptions $\hat{c}$ along the way, so that at each time the proof contains a single encrypted bit.

We now describe the bounded-space SNARK $(\mathbb{G}, \mathbb{P}, \mathbb{V})$ in detail, focusing mostly on the differences from the case of public verification, which will often appear in boxes. Once again, the main technical component in the construction is the recursive-verification machine $\mathbb{M}$, defined next.

The recursive-verification machine $\mathbb{M}$. Let τ be a (private) verification state for a SNARK for $\mathbf{NP}$, corresponding to security parameter n . Let $y = (\mathcal{M}, x, t) \in \mathcal{L}_c$, where $\mathcal{M}$ is a read-once machine (Definition 113) with bounded space s , and $|y| \leq 4n$. Let $\boxed{c^\tau}$ be an encryption of τ .

The recursive-verification machine $\mathbb{M}^{y, \boxed{\mathbf{c}^\tau}}$ takes input χ and witness ω . The input χ is a tuple $(\mathbf{i}, \mu, \boxed{\hat{\mathbf{c}}_{\mathbf{i}}}, \mathbf{S}_{\mathbf{i}}, \mathbf{S}_{\mathbf{i}+n^j})$, where as in the case of public verification, μ is a description of a machine, $t = n^d$, $\mathbf{i} \in [n]^{d-j}$, for some $j \in \{0, \dots, d\}$, and $\mathbf{S}_{\mathbf{i}}, \mathbf{S}_{\mathbf{i}+n^j} \in \{0, 1\}^s$. In addition, the input now includes an $\boxed{\hat{\mathbf{c}}}$ encryption of a bit. The witness ω is a tuple $(\vec{\pi}, \boxed{\vec{\hat{\mathbf{c}}}}, \vec{\mathbf{S}}, w)$, where as in the public verification case, $\pi = (\pi_{\mathbf{i}+n^{j-1}}, \pi_{\mathbf{i}+2n^{j-1}}, \dots, \pi_{\mathbf{i}+n^j}) \in \{0, 1\}^{p(n) \times n}$ is a vector of n SNARK proofs, $\vec{\mathbf{S}} = (\mathbf{S}_{\mathbf{i}+n^{j-1}}, \mathbf{S}_{\mathbf{i}+2n^{j-1}}, \dots, \mathbf{S}_{\mathbf{i}+(n-1)n^{j-1}}) \in \{0, 1\}^{s \times (n-1)}$ is a vector of $n-1$ intermediate states, and $w \in \{0, 1\}$ is a witness bit. In addition the witness now includes a vector $\boxed{\vec{\hat{\mathbf{c}}}}$ of n bit encryptions.

$\mathbb{M}^{y, \mathbf{c}^\tau}$ recursively verifies the correctness of the computation given by y :

- **The halting condition:** for any leaf $\mathbf{i} \in [n]^d$, the machine verifies the correctness of the step specified by the leaf. That is, it verifies that $\widetilde{\mathcal{M}}_x(\mathbf{S}_{\mathbf{i}}, w) = \mathbf{S}_{\mathbf{i}+1}$.
- **The recursive step:** in any other node $\mathbf{i} \in [n]^{d-j}, j \in [d]$, the machine $\boxed{\text{homomorphically}}$ runs the SNARK verifier $\mathcal{V}$ on $\mathbf{c}^\tau$ for the proofs in $\vec{\pi}$ attesting to the correctness of the sub-computation corresponding to the subtree rooted at $\mathbf{i}$. It homomorphically multiplies the resulting verification bits, and compares to the input encryption bit $\hat{\mathbf{c}}$.

The formal description of $\mathbb{M}^{y, \mathbf{c}^\tau}$ is given in Figure 5.5.

Parameter setting. The constant $\bar{c}$ determining the relation $\mathcal{R}_{\bar{c}}$, and the time bound $t_{\mathcal{M}, \mathcal{V}} = t_{\mathcal{M}, \mathcal{V}}(n, S)$, referred to in Figure 5.5, are chosen similarly to the case of public verification. The additional homomorphic operations incur a multiplicative polynomial overhead in the security parameter n .

The bounded-space SNARK. We now describe each components of the space-bounded SNARK $(\mathbb{G}, \mathbb{P}, \mathbb{V})$ for read once computations in $\mathcal{L}_c$, for any $c \in \mathbb{N}$, based on a designated-verifier preprocessing SNARK $(\mathcal{G}, \mathcal{P}, \mathcal{V})$ for $\mathcal{R}_{\bar{c}}$. The description is provided in Figure 5.6. As before, we shall consider a computation tree of arity n , and depth d where nodes are labeled by $\{\mathbf{i} \in [n]^j : 0 \leq j \leq d\}$. Also as above, for every leaf $\mathbf{i} \in [n]^d$, we denote by $\underline{\mathbf{i}}$ the corresponding number in $\{0, 1, \dots, n\}$, and for $\mathbf{i} \in [n]^{d-j}$, we denote by $\underline{\mathbf{i}}$ the number corresponding to its left most descendant leaf $(\mathbf{i}, 1^j)$.

Setting: Let $\mathcal{V}$ be a SNARK verifier for $\mathcal{R}_{\bar{c}}$ (the choice of $\bar{c}$ is addressed in the *parameter setting* paragraph below). Let τ be a (private) verification state for $\mathcal{V}$, corresponding to security parameter 1^n , and let $\boxed{\mathbf{c}^\tau}$ be an encryption of τ . Let $y = (\mathcal{M}, x, t)$, where $\mathcal{M}$ is a read-once machine, with bounded space $s \leq S \leq t \leq n^{\log n}$, and transition circuit $\widetilde{\mathcal{M}}_x$. Let d be such that $t = n^d$.

$\mathbb{M}^{y, \mathbf{c}^\tau}(\chi, \omega)$:

1. **Parsing input and witness:** Parse χ as $(\mathbf{i}, \mu, \boxed{\hat{\mathbf{c}}_1}, S, S')$, and ω as $(\vec{\pi}, \boxed{\vec{\hat{\mathbf{c}}}}, \vec{S}, w)$.
2. **Base case:** If $\mathbf{i} \in [n]^d$, verify that $\widetilde{\mathcal{M}}_x(S, w) = S'$.
3. **Recursive case:** If $\mathbf{i} \in [n]^{d-j}$, for some $j \in [d]$, then
 - Parse $\vec{\pi}$ as $(\pi_1, \dots, \pi_n)$, $\boxed{\vec{\hat{\mathbf{c}}}}$ as $(\hat{\mathbf{c}}_1, \dots, \hat{\mathbf{c}}_n)$, and $(S, \vec{S}, S')$ as $(S_0, S_1, \dots, S_n)$.
 - For each $k \in [n]$, homomorphically compute

$$\boxed{\hat{\mathbf{c}}_k^\mathcal{V} = \text{Eval} \left(\mathcal{V}_{\bar{c}} \left(\cdot, (\mu, \chi_k, t_{\mathcal{M}, \mathcal{V}}), \pi_k \right), \mathbf{c}^\tau \right)},$$

where $\chi_k = ((\mathbf{i}, k), \mu, \boxed{\hat{\mathbf{c}}_k}, S_{k-1}, S_k)$, and $t_{\mathcal{M}, \mathcal{V}}$ is a fixed polynomial in (n, S) , whose choice is addressed in the *parameter setting* paragraph below.

- Verify that

$$\boxed{\hat{\mathbf{c}}_1 = \text{Eval} \left(\prod_{k \in [n]} (\cdot, \cdot), (\hat{\mathbf{c}}_k^\mathcal{V}, \hat{\mathbf{c}}_k) \right)};$$

if $\hat{\mathbf{c}}_k = \perp$, do not include it in the above homomorphic product.

Figure 5.5: The recursive-verification machine $\mathbb{M}^{y, \mathbf{c}^\tau}$.

Proof of security. We now show that $(\mathbb{G}, \mathbb{P}, \mathbb{V})$ is a bounded-space (designated-verifier) SNARK for read-once computations in $\mathcal{R}_c$. The completeness and efficiency properties of the system immediately follow from those of the SNARK $(\mathcal{G}, \mathcal{P}, \mathcal{V})$, and from the correctness of the FHE scheme. We concentrate on proving the adaptive proof of knowledge property.

Recall that for any $c \in \mathbb{N}$, and any (malicious) polysize prover $\mathbb{P}^*$, we would like to construct a corresponding polysize extractor $\mathbb{E}_{\mathbb{P}^*}$ such that, when $\mathbb{P}^*$ convinces $\mathbb{V}$ of a statement $y = (\mathcal{M}, x, t)$, where $|y| \leq 4n$, $t \leq |x|^c$ and $\mathcal{M}$ uses space at most S , the extractor can find a witness $w \in \{0, 1\}^t$, such that $(y, w) \in \mathcal{R}_c$, with all but negligible probability. As in the case of public verification, to achieve this goal, we employ a recursive extraction strategy, which we now describe.

Given the prover $\mathbb{P}^*$, we construct c (families of) polysize extractors $\mathcal{E}_1, \dots, \mathcal{E}_c$, one

for each potential level in the computation tree. As before, to make notation lighter, we do not explicitly write the auxiliary input z that may be given to $\mathbb{P}^*$ and its extractor $\mathbb{E}_{\mathbb{P}^*}$.

The extractors are constructed as follows:

- Use the prover $\mathbb{P}^*$ to construct a SNARK prover $\mathcal{P}_1^*$ that works as follows: on input (σ, c^τ) , $\mathcal{P}_1^*$ computes $(y, \pi_\varepsilon, \hat{c}_\varepsilon) \leftarrow \mathbb{P}^*(\sigma, c^\tau)$, constructs the instance $y_\varepsilon := (\mathbb{M}^{y, c^\tau}, (\varepsilon, \hat{c}_\varepsilon, \mathbb{M}^{y, c^\tau}, \text{init}, \text{acc}), t_{\mathcal{M}, \mathcal{V}})$, and outputs $(y_\varepsilon, \pi_\varepsilon)$.

Then define $\mathcal{E}_1 := \mathcal{E}_{\mathcal{P}_1^*}$ to be the SNARK extractor for the SNARK prover $\mathcal{P}_1^*$. Like $\mathcal{P}_1^*$, $\mathcal{E}_1$ also expects input (σ, c^τ) ; $\mathcal{E}_1$ returns $(\vec{\pi}, \vec{\hat{c}}, \vec{S}, w)$ that is (with all but negligible probability) a valid witness for the SNARK statement y_ε , assuming that $\mathbb{V}$ (and thus also $\mathcal{V}$) accepts π_ε .

- Use $\mathcal{E}_1$ to construct the new SNARK prover $\mathcal{P}_2^*$ that works as follows: on input (σ, c^τ) , $\mathcal{P}_2^*$ computes $(\vec{\pi}, \vec{\hat{c}}, \vec{S}, w) \leftarrow \mathcal{E}_1(\sigma, c^\tau)$ and then outputs $(\vec{y}, \vec{\pi})$, where the vector of SNARK statements $\vec{y}$ contains an entry $y_i := (\mathbb{M}^{y, c^\tau}, (i, \hat{c}_i, \mathbb{M}^{y, c^\tau}, S_{i-1}, S_i), t_{\mathcal{M}, \mathcal{V}})$ for each $i \in [n]$. Then define $\mathcal{E}_2 := \mathcal{E}_{\mathcal{P}_2^*}$ to be the SNARK extractor for the SNARK prover $\mathcal{P}_2^*$. Given (σ, c^τ) (with all but negligible probability) $\mathcal{E}_2$ should output a witness for each accepting statement and convincing proof (y_i, π_i) in $(\vec{y}, \vec{\pi})$.
- In general, for each $1 < j \leq c$, we similarly define $\mathcal{P}_j^*$ that computes $\left\{ (\vec{\pi}^{(i)}, \vec{\hat{c}}^{(i)}, \vec{S}^{(i)}, w^{(i)}) : i \in [n]^{j-1} \right\} \leftarrow \mathcal{E}_{j-1}(\sigma, c^\tau)$ and then outputs $\left\{ (\vec{y}^{(i)}, \vec{\pi}^{(i)}) : i \in [n]^{j-1} \right\}$, where the vector of SNARK statements $\vec{y}^{(i)}$ contains an entry $y_{(i, k)} := (\mathbb{M}^{y, c^\tau}, ((i, k), \hat{c}_{(i, k)}, \mathbb{M}^{y, c^\tau}, S_{k-1}^{(i)}, S_k^{(i)}), t_{\mathcal{M}, \mathcal{V}})$ for each $k \in [n]$. Then define $\mathcal{E}_j := \mathcal{E}_{\mathcal{P}_j^*}$ to be the SNARK extractor for the SNARK prover $\mathcal{P}_j^*$.

We now define the extractor $\mathbb{E}_{\mathbb{P}^*}$. On input (σ, c^τ) , $\mathbb{E}_{\mathbb{P}^*}$ runs $\mathbb{P}^*(\sigma, c^\tau)$ and obtains the instance $y = (\mathcal{M}, x, t)$. It then computes $d := \log_n t$. If $d > c$ $\mathbb{E}_{\mathbb{P}^*}$ aborts, and otherwise it runs $\mathcal{E}_d(\sigma, c^\tau)$ and obtains $\left\{ (\vec{\pi}^{(i)}, \vec{\hat{c}}^{(i)}, \vec{S}^{(i)}, w^{(i)}) : i \in [n]^d \right\}$. It then outputs as the witness $w = \{w^{(i)} : i \in [n]^d\}$ as the witness for y .

Note that each $\mathcal{E}_j$ is of polynomial size, because each $\mathcal{E}_j$ is constructed via a constant number of recursive invocations of the SNARK proof of knowledge, and each such invocation incurs an arbitrary polynomial blowup in the size of the extractor relative to its prover. Thus, we deduce that $\mathbb{E}_{\mathbb{P}^*}$ is also of polynomial size.

It is left to argue that w extracted by $\mathbb{E}_{\mathbb{P}^*}$ is a valid witness for y :

Proposition 123. *Let $\mathbb{P}^*$ be a polysize prover for the bounded-space SNARK, and let $\mathbb{E}_{\mathbb{P}^*}$ be its corresponding extractor as defined above. Then for any $n \in \mathbb{N}$, and space bound $S \in \mathbb{N}$:*

$$\Pr \left[\begin{array}{c} \mathbb{V}_c((\tau, \mathbf{sk}), y, \pi) = 1 \\ (y, w) \notin \mathcal{R}_c \end{array} \middle| \begin{array}{c} ((\sigma, \mathbf{c}^\tau), (\tau, \mathbf{sk})) \leftarrow \mathbb{G}(1^n, S) \\ (y, \pi) \leftarrow \mathbb{P}^*(\sigma, \mathbf{c}^\tau) \\ w \leftarrow \mathbb{E}_{\mathbb{P}^*}(\sigma, \mathbf{c}^\tau) \end{array} \right] \leq \text{negl}(n) .$$

Proof. We assume throughout that $|y| \leq 4n$, $t = n^d$, for $d \leq c$, and that $\mathcal{M}$ is a read-once machine that uses space at most S ; indeed, $\mathbb{V}$ explicitly checks all of these conditions. For each node $\mathbf{i} \in [n]^j, j \in \{0, \dots, d\}$, we consider the corresponding input $\chi_{\mathbf{i}} := (\mathbb{M}^{y, \mathbf{c}^\tau}, (\mathbf{i}, \hat{\mathbf{c}}_{\mathbf{i}}, \mathbb{M}^{y, \mathbf{c}^\tau}, \mathbf{S}_{\mathbf{i}}, \mathbf{S}_{\mathbf{i}+n^{d-|\mathbf{i}|}}), t_{\mathcal{M}, \nu})$ and witness $\omega_{\mathbf{i}} = (\vec{\pi}^{(\mathbf{i})}, \vec{\hat{\mathbf{c}}}^{(\mathbf{i})}, \vec{\mathbf{S}}^{(\mathbf{i})}, w^{(\mathbf{i})})$ induced by the recursive extraction procedure, and show that $\mathbb{M}^{y, \mathbf{c}^\tau}(\chi_{\mathbf{i}}, \omega_{\mathbf{i}}) = 1$. If this is the case, then by the construction of $\mathbb{M}$, the extracted $\{\mathbf{S}_{\mathbf{i}}, \mathbf{S}_{\mathbf{i}+1}, w^{(\mathbf{i})} : \mathbf{i} \in [n]^d\}$, satisfy

$$S_0 = \text{init}, S_{\mathbf{i}} = \widetilde{\mathcal{M}}_x(S_{\mathbf{i}-1}, w^{(\mathbf{i})}), S_t = \text{acc} ,$$

Implying that $\mathcal{M}(y, w) = 1$.

The fact that $\mathbb{M}^{y, \mathbf{c}^\tau}(\chi_{\mathbf{i}}, \omega_{\mathbf{i}}) = 1$ is shown by induction on the level of the computation tree. For the base case, we show that, except with negligible probability, whenever the prover $\mathbb{P}^*$ convinces the verifier $\mathbb{V}$ to accept $(y, \pi_\varepsilon, \hat{\mathbf{c}}_\varepsilon)$, the extractor $\mathcal{E}_1$ outputs $\omega_\varepsilon = (\vec{\pi}, \vec{\hat{\mathbf{c}}}, \vec{\mathbf{S}}, w)$ such that letting $\chi_i = (i, \hat{\mathbf{c}}_i, \mathbb{M}^{y, \mathbf{c}^\tau}, \mathbf{S}_{i-1}, \mathbf{S}_i)$, $\hat{\mathbf{c}}_i^\nu = \text{Eval}\left(\mathcal{V}_{\hat{\mathbf{c}}}(\cdot, (\mathbb{M}^{y, \mathbf{c}^\tau}, \chi_i, t_{\mathcal{M}, \nu}), \pi_i), \mathbf{c}^\tau\right)$, $y_i := (\mathbb{M}^{y, \mathbf{c}^\tau}, \chi_i, t_{\mathcal{M}, \nu})$, it holds that:

1. $\hat{\mathbf{c}}_\varepsilon = \text{Eval}\left(\prod_{i \in [n]}, (\hat{\mathbf{c}}_i^\nu, \hat{\mathbf{c}}_i)\right)$,
2. for each $i \in [n]$, $\text{Dec}_{\mathbf{sk}}(\hat{\mathbf{c}}_i) = 1$,
3. for each $i \in [n]$, $\mathcal{V}_{\hat{\mathbf{c}}}(\tau, y_i, \pi_i) = 1$.

First, the fact that $\mathbb{V}_c((\tau, \mathbf{sk}), y, (\pi_\varepsilon, \hat{\mathbf{c}}_\varepsilon)) = 1$ implies that $\mathcal{V}_{\hat{\mathbf{c}}}(\tau, y_\varepsilon, \pi_\varepsilon) = 1$, where $y_\varepsilon = (\mathbb{M}^{y, \mathbf{c}^\tau}, (\varepsilon, \hat{\mathbf{c}}_\varepsilon, \mathbb{M}^{y, \mathbf{c}^\tau}, \text{init}, \text{acc}), t_{\mathcal{M}, \nu})$. Next, we would like to invoke the SNARK proof of knowledge property; however, we need to deal with the fact that the prover gets as extra auxiliary input an encryption $\mathbf{c}^\tau$ of the secret verification state τ . We invoke semantic security: we first consider an alternative experiment where the prover get an encryption $\mathbf{c}^0$ of $0^{|\tau|}$ rather than of τ .

We first argue that, in the alternative experiment, whenever $\mathbb{V}$ accepts (and except with negligible probability), the first of the three conditions above must hold; namely, $\hat{\mathbf{c}}_\varepsilon = \text{Eval}\left(\prod_{i \in [n]}, (\hat{\mathbf{c}}_i^\mathcal{V}, \hat{\mathbf{c}}_i)\right)$, where now $\mathbf{c}^\tau$ is replaced everywhere with $\mathbf{c}^0$. Indeed, in the alternative experiment, the SNARK prover $\mathcal{P}_1^*$ is only given the auxiliary input $\mathbf{c}^0$, which is independent of the verification state τ ; hence, the SNARK proof of knowledge can be invoked. Specifically, except with negligible probability, whenever $\mathbb{V}$ (and hence also $\mathcal{V}$) accepts, it must be that the extractor $\mathcal{E}_1$ outputs a valid witness $(\vec{\pi}, \vec{\tilde{\mathbf{c}}}, \vec{\mathbf{S}}, w)$ for the statement $y_\varepsilon = (\mathbb{M}^{y, \mathbf{c}^0}, (\varepsilon, \hat{\mathbf{c}}_\varepsilon, \mathbb{M}^{y, \mathbf{c}^0}, \text{init}, \text{acc}), t_{\mathcal{M}, \mathcal{V}})$ output by $\mathcal{P}_1^*$ (when given $\mathbf{c}^0$ rather than $\mathbf{c}^\tau$).

In particular, by construction of $\mathbb{M}$, we deduce that $(\vec{\pi}, \vec{\tilde{\mathbf{c}}}, \vec{\mathbf{S}}, w)$ satisfies the first of the three conditions above. Next, note this first condition can be efficiently tested given only $(\omega_\varepsilon, \mathbf{c}^0)$, by running the (deterministic) algorithm $\mathcal{V}, \text{Eval}$. In particular, neither $\mathbf{sk}$ nor τ are required for such a test. We can thus deduce that in the original experiment, where $\mathcal{P}_1^*$ and $\mathcal{E}_1$ are given $\mathbf{c}^\tau$, the first condition holds with all but negligible probability.

We now note that, whenever $\mathbb{V}$ accepts, $\text{Dec}_{\mathbf{sk}}(\hat{\mathbf{c}}_\varepsilon) = 1$, thus the fact that the first condition holds, together with the correctness of homomorphic evaluation, implies that the last two of the three conditions must also hold. Indeed, since each one of the ciphers in $\vec{\tilde{\mathbf{c}}}$, and $\vec{\tilde{\mathbf{c}}}^\mathcal{V}$ must also decrypt to 1, or their homomorphic product would decrypt to 0 rather than 1. This already implies the second condition. By the way that $\vec{\tilde{\mathbf{c}}}^\mathcal{V}$ is computed, the third condition holds, i.e., for each $i \in [n]$, $\mathcal{V}_{\tilde{\mathbf{c}}}(\tau, y_i, \pi_i) = 1$.

To complete the proof, we can prove in a similar manner the inductive step. That is, assuming that the two last of the three conditions are satisfied at the j -th level of the tree, we can deduce that the first condition holds for level $j + 1$. This is done by first establishing condition one in an alternative experiment where $\mathbf{c}^\tau$ is replaced by $\mathbf{c}^0$, and then invoking semantic security to deduce the same for the original experiment. We then deduce, from the first condition, the last two conditions as we did for the base case.

This concludes the proof of Proposition 123. $\square$

5.4.3 Complexity Preservation

In this section, we observe that the SNARKs constructed satisfy essentially optimal efficiency properties, meaning that each of the components $\mathcal{G}, \mathcal{P}, \mathcal{V}$ has the best possible asymptotic complexity up to multiplicative factors in the security parameter. We call

such SNARKs *complexity preserving*.

Definition 124. A triple of algorithms $(\mathcal{G}, \mathcal{P}, \mathcal{V})$ is a **complexity-preserving SNARK for NP** if it is a fully-succinct SNARK for NP with a stronger efficiency guarantee: There exists a universal polynomial p such that, for every $c \in \mathbb{N}$, every large enough security parameter $n \in \mathbb{N}$, and $y = (\mathcal{M}, x, t) \in \mathcal{R}_c$,

- the generator $\mathcal{G}(1^n)$ runs in time $p(n)$;
- the prover $\mathcal{P}(\sigma, y, w)$ runs in time $(|\mathcal{M}| + |x| + t) \cdot p(n)$;
- the prover $\mathcal{P}(\sigma, y, w)$ runs in space $(|\mathcal{M}| + |x| + s) \cdot p(n)$, where s is a bound on the space used by $\mathcal{M}$;
- the verifier $\mathcal{V}_c(\tau, y, \pi)$ runs in time $(|\mathcal{M}| + |x|) \cdot p(n)$;
- an honestly generated proof has size $p(n)$.

Theorem 125 (Bootstrapping SNARKs - restated). *Assuming collision-resistant hash functions any SNARK can be transformed into a complexity-preserving SNARK.*

Proof sketch. We claim that the same bootstrapping procedure described in Sections 5.4.1, 5.4.2 and summarized in Figure 5.4, with a small change to the prover's strategy described in Sections 5.4.1.2, 5.4.2.1 is already complexity preserving. We briefly recall the complexity of each component, and in particular note the slight change required for the prover.

The prover. For simplicity of exposition, in Sections 5.4.1.2, 5.4.2.1 the prover $\mathbb{P}$ of the bounded-space SNARK constructs the computation tree, and augments it with SNARK proofs, level by level. This, in particular, causes the space used by the prover to be as large as the computation time t .

Instead, we could have the prover $\mathbb{P}$ gradually construct the tree of proofs along with the computation, and delete SNARK proofs and other information that is no longer relevant as it proceeds. Specifically, the prover is going to store the information, including SNARK proofs and statements, corresponding to at most n nodes, at each level of the computation tree and at each point in time. Whenever n consecutive proofs and statements are obtained, the prover can aggregate them into a single proof in the level above, and delete all the corresponding information. Overall, since there are only

c levels, the space used by the prover is at most $S \cdot \text{poly}(n)$, where S is the bound on the space of the underlying space-bounded machine, which after the transformations in Section 5.4.1.3 is $\text{poly}(n)$.

The above does not count, however, the space required to produce the witness w' produced by the transformations in Section 5.4.1.3, which is $s \cdot \text{poly}(n)$, where s is the bound on the space used by the original $\mathcal{M}$.

The generator. Since the SNARK is fully-succinct the generator runs in time $\text{poly}(n)$ as required.

The verifier. The verifier first reduces its instance y to an instance y_1 of size $O(n)$, by applying the first transformation from Section 5.4.1.3, which requires time $|y| \cdot \text{poly}(n)$. It then applies the second transformation and the bounded-space SNARK, both taking time $\text{poly}(|y_1|, n)$, and thus overall the running time of the verifier is $|y| \cdot \text{poly}(n)$.

The length of the proof. The proof produced by the prover is, in fact, produced by the prover of the bounded-space SNARK, and is thus guaranteed to be of size $\text{poly}(n)$. □

- **The generator.** Given the security parameter 1^n , and space bound S , the generator $\mathbb{G}$, computes $B = \text{poly}_{\mathcal{M}, \mathcal{V}}(n, S)$, where $\text{poly}_{\mathcal{M}, \mathcal{V}}$ is computed as described above, runs the preprocessing SNARK generator $\mathcal{G}(1^n, B)$ in order to obtain a reference string σ and (private) verification state τ . In addition, it samples a secret key sk for an FHE scheme, and computes an encryption $\text{c}^\tau \leftarrow \text{Enc}_{\text{sk}}(\tau)$. Finally, it outputs $(\tilde{\tau}, \tilde{\sigma}) = ((\tau, \text{sk})(\sigma, \text{c}^\tau))$.
- **The prover.** Given $y = ((\mathcal{M}, x, t), w) \in \mathcal{R}_c$, where $|y| \leq 4n$, $w \in \{0, 1\}^t$ and $t = n^d$ for some $d \in \mathbb{N}$, the prover $\mathbb{P}$ gradually constructs the computation tree augmented with SNARK proofs attesting to the correctness of the computation given by y :

- For a leaf $\mathbf{i} \in [n]^d \setminus \{1^d\}$:
 1. Given state $\mathbf{S}_{\mathbf{i}-1}$ of $\mathcal{M}(x, w)$, compute the next state $\mathbf{S}_{\mathbf{i}} = \widetilde{\mathcal{M}}_x(\mathbf{S}_{\mathbf{i}-1}, w_{\mathbf{i}})$.
 2. Invoke $\mathcal{P}(\sigma, \dots)$ to compute a SNARK proof $\pi_{\mathbf{i}}$ for $(y_{\mathbf{i}}, \omega_{\mathbf{i}}) \in \mathcal{R}_{\bar{c}}$, where $y_{\mathbf{i}} = (\mathbb{M}^{y, \text{c}^\tau}, \chi_{\mathbf{i}}, t_{\mathcal{M}, \mathcal{V}})$, $\chi_{\mathbf{i}} = (\mathbf{i}, \hat{\mathbf{c}}_{\mathbf{i}}, \mathbb{M}^{y, \text{c}^\tau}, \mathbf{S}_{\mathbf{i}-1}, \mathbf{S}_{\mathbf{i}}, t_{\mathcal{M}, \mathcal{V}})$, $\omega_{\mathbf{i}} = (\perp, \perp, \perp, w_{\mathbf{i}})$, $\hat{\mathbf{c}}_{\mathbf{i}} = \perp$, and $\bar{c}$, $t_{\mathcal{M}, \mathcal{V}} = t_{\mathcal{M}, \mathcal{V}}(n, S)$ were specified in the *parameter setting* paragraph.
- For a node $\mathbf{i} \in [n]^{d-j}$, where $j \in [d]$,
 1. Let $(\mathbf{S}, \vec{\mathbf{S}}, \mathbf{S}') = (\mathbf{S}_{\mathbf{i}}, \mathbf{S}_{(\mathbf{i}, 1)}, \dots, \mathbf{S}_{(\mathbf{i}, n-1)}, \mathbf{S}_{(\mathbf{i}, n)})$, where $\mathbf{S}_{(\mathbf{i}, k)} = \mathbf{S}_{\mathbf{i} + kn^j}$.
 2. Let $\vec{\pi} = (\pi_{(\mathbf{i}, 1)}, \pi_{(\mathbf{i}, 2)}, \dots, \pi_{(\mathbf{i}, n)})$ and $\vec{\hat{\mathbf{c}}} = (\hat{\mathbf{c}}_{(\mathbf{i}, 1)}, \hat{\mathbf{c}}_{(\mathbf{i}, 2)}, \dots, \hat{\mathbf{c}}_{(\mathbf{i}, n)})$.
 3. For $k \in [n]$ homomorphically compute

$$\hat{\mathbf{c}}_{(\mathbf{i}, k)}^\mathcal{V} = \text{Eval} \left(\mathcal{V}_{\bar{c}} \left(\cdot, (\mathbb{M}^{y, \text{c}^\tau}, \chi_{(\mathbf{i}, k)}, t_{\mathcal{M}, \mathcal{V}}), \pi_{(\mathbf{i}, k)} \right), \text{c}^\tau \right),$$

where $\chi_k = ((\mathbf{i}, k), \mathbb{M}^{y, \text{c}^\tau}, \hat{\mathbf{c}}_{(\mathbf{i}, k)}, \mathbf{S}_{(\mathbf{i}, k-1)}, \mathbf{S}_{(\mathbf{i}, k)})$, and $t_{\mathcal{M}, \mathcal{V}}$ is a fixed polynomial in (n, S) , addressed in the *parameter setting* paragraph.

- 4. Compute

$$\hat{\mathbf{c}}_{\mathbf{i}} = \text{Eval} \left(\prod_{k \in [n]} (\cdot, \cdot), (\hat{\mathbf{c}}_{(\mathbf{i}, k)}^\mathcal{V}, \hat{\mathbf{c}}_{(\mathbf{i}, k)}) \right);$$

if $\hat{\mathbf{c}}_{(\mathbf{i}, k)} = \perp$, do not include it in the above homomorphic product.

- 5. Invoke $\mathcal{P}(\sigma, \dots)$ to compute a SNARK proof $\pi_{\mathbf{i}}$ for $(y_{\mathbf{i}}, \omega_{\mathbf{i}}) \in \mathcal{R}_{\bar{c}}$, where $y_{\mathbf{i}} = (\mathbb{M}^{y, \text{c}^\tau}, \chi_{\mathbf{i}}, t_{\mathcal{M}, \mathcal{V}})$, $\chi_{\mathbf{i}} = (\mathbf{i}, \hat{\mathbf{c}}_{\mathbf{i}}, \mathbb{M}^{y, \text{c}^\tau}, \mathbf{S}, \mathbf{S}', t_{\mathcal{M}, \mathcal{V}})$, $\omega_{\mathbf{i}} = (\vec{\pi}, \vec{\hat{\mathbf{c}}}, \vec{\mathbf{S}}, \perp)$.

- The final proof is $(\pi_\varepsilon, \hat{\mathbf{c}}_\varepsilon)$ corresponding to the root ε .

- **The verifier.** Given $c \in \mathbb{N}$, $y = (\mathcal{M}, x, t)$, and proof $(\pi_\varepsilon, \hat{\mathbf{c}}_\varepsilon)$,

1. Verify that $|y| \leq 4n$, $t \leq |x|^c$, and that $\mathcal{M}$ uses space at most $S \leq t$.^a
2. Verify that:

$$\mathcal{V}_{\bar{c}} \left(\tau, (\mathbb{M}^{y, \text{c}^\tau}, \chi_\varepsilon, t_{\mathcal{M}, \mathcal{V}}), \pi_\varepsilon \right) = 1 \quad \bigwedge \quad \text{Dec}_{\text{sk}}(\hat{\mathbf{c}}_\varepsilon) = 1,$$

where $\chi_\varepsilon = (\varepsilon, \hat{\mathbf{c}}_\varepsilon, \mathbb{M}^{y, \text{c}^\tau}, \text{init}, \text{acc})$, init and acc represent the initial state and final accepting state of $\mathcal{M}$ (respectively), and $t_{\mathcal{M}, \mathcal{V}} = t_{\mathcal{M}, \mathcal{V}}(n, S)$ were specified in the *parameter setting* paragraph.

^aWe insist that there is an explicit space bound in $\mathcal{M}$'s description.

Figure 5.6: A bounded-space (designated-verifier) SNARK for read-once computations in NP

Chapter 6

Conclusion and Future Research

In this thesis, we have explored new approaches towards extraction of knowledge from adversarial code. A connecting thread, present in most of our results, is the contrast between code obfuscation and knowledge extraction. Indeed, the goal of obfuscation is exactly to foil any efficient extraction of significant knowledge from the code.

While we have exhibited several new ways of extracting knowledge from adversarial code, there is still a long way towards a more complete understanding of the knowledge vs. obfuscation tradeoff. For once, the recent burst of research and candidate constructions for obfuscation may lead to a dramatic change in our understanding of this question. Indeed, for the first time after many years, there is evidence that perhaps there is a general way to immunize programs against efficient knowledge extraction. Still, the existence of various types of extractable, or unobfuscatable, functions sets certain limits on the reach of such an obfuscation procedure. Thus, there is room to further enhance and leverage on knowledge extraction techniques; as we have seen, the ability to extract from specific (or contrived) programs may just be enough to get any given application.

Another point to be noted is that code obfuscation does not say anything about the possibility of *non-uniform* (or *non-universal*) extraction. Indeed, when trying to extract from an adversarial code, it may be sufficient to be able to extract only given appropriate non-uniform advice that may be inefficiently derived from the code of the adversary. This inefficient derivation may very well “deobfuscate” the adversary.

In any case, future progress in either direction, obfuscation or extraction, would shed new light on the other.

Bibliography

- [1] Galileo's anagrams and the moons of mars. Math Pages. <http://www.mathpages.com/home/kmath151/kmath151.htm>. 2
- [2] Taki. Wikipedia. [http://en.wikipedia.org/wiki/Taki_\(card_game\)](http://en.wikipedia.org/wiki/Taki_(card_game)). (document)
- [3] Masayuki Abe and Serge Fehr. Perfect NIZK with adaptive soundness. In *Proceedings of the 4th Theory of Cryptography Conference*, TCC '07, pages 118–136, 2007. 4.1.6
- [4] William Aiello, Sandeep N. Bhatt, Rafail Ostrovsky, and Sivaramakrishnan Rajagopalan. Fast verification of any remote procedure call: Short witness-indistinguishable one-round proofs for NP. In *Proceedings of the 27th International Colloquium on Automata, Languages and Programming*, ICALP '00, pages 463–474, 2000. 5.1.1, 5.1.5
- [5] William Aiello, Yuval Ishai, and Omer Reingold. Priced oblivious transfer: How to sell digital goods. In *EUROCRYPT*, pages 119–135, 2001. 4.5.3
- [6] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. From secrecy to soundness: Efficient verification via secure computation. In *Proceedings of the 37th International Colloquium on Automata, Languages and Programming*, ICALP '10, pages 152–163, 2010. 5.1.1
- [7] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and the hardness of approximation problems. *Journal of the ACM*, 45(3):501–555, 1998. Preliminary version in FOCS '92. 1.1
- [8] Sanjeev Arora and Shmuel Safra. Probabilistic checking of proofs: a new characterization of NP. *Journal of the ACM*, 45(1):70–122, 1998. Preliminary version in FOCS '92.
- [9] László Babai, Lance Fortnow, Leonid A. Levin, and Mario Szegedy. Checking computations in polylogarithmic time. In *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing*, STOC '91, pages 21–32, 1991. 5.1.1
- [10] László Babai, Lance Fortnow, and Carsten Lund. Nondeterministic exponential time has two-prover interactive protocols. In *Proceedings of the 31st Annual Symposium on Foundations of Computer Science*, SFCS '90, pages 16–25, 1990. 1.1
- [11] Boaz Barak. How to go beyond the black-box simulation barrier. In *FOCS*, pages 106–115, 2001. 1, 1.1, 3.1.1, 4.1.4

- [12] Boaz Barak. Constant-round coin-tossing with a man in the middle or realizing the shared random string model. In *FOCS*, pages 345–355, 2002. [3.1.1](#)
- [13] Boaz Barak, Sanjam Garg, Yael Tauman Kalai, Omer Paneth, and Amit Sahai. Protecting obfuscation against algebraic attacks. In *EUROCRYPT*, pages 221–238, 2014. [1](#), [4.3.1](#)
- [14] Boaz Barak and Oded Goldreich. Universal arguments and their applications. *SIAM J. Comput.*, 38(5):1661–1694, 2008. [3.1.4](#), [4.4.1](#), [5.2](#)
- [15] Boaz Barak, Oded Goldreich, Shafi Goldwasser, and Yehuda Lindell. Resetably-sound zero-knowledge and its applications. In *FOCS*, pages 116–125, 2001. [1.1](#), [3.1.1](#), [3.1.1](#), [3.1.1](#), [2](#), [3.1.5](#), [3.1.5](#), [3.1.5](#), [3.1.7](#), [3.5.1](#), [3.5.1](#), [39](#), [3.5.1](#), [3.5.1](#), [3.5.1](#), [3.5.5](#), [49](#), [3.5.5](#), [3.5.6](#)
- [16] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In *CRYPTO*, pages 1–18, 2001. [1](#), [3.1.2](#), [3.1.3](#), [3.1.4](#), [3.1.6](#), [3.1.6](#), [3.1.6](#), [3.1.7](#), [14](#), [16](#), [3.3.1](#), [24](#), [3.3.1](#), [3.3.3](#), [31](#), [3.6](#), [4.1.3](#), [4.1.3](#), [4.1.6](#), [4.3.1](#), [64](#)
- [17] Boaz Barak and Yehuda Lindell. Strict polynomial-time in simulation and extraction. In *STOC*, pages 484–493, 2002. [3.1.1](#)
- [18] Boaz Barak, Yehuda Lindell, and Salil P. Vadhan. Lower bounds for non-black-box zero knowledge. *J. Comput. Syst. Sci.*, 72(2):321–391, 2006. [4.1.4](#), [4.1.5](#), [4.5.1](#)
- [19] Boaz Barak and Amit Sahai. How to play almost any mental game over the net - concurrent composition via super-polynomial simulation. *Electronic Colloquium on Computational Complexity (ECCC)*, (096), 2005. [3.1.1](#), [5.3.2.3](#)
- [20] Mihir Bellare and Oded Goldreich. On defining proofs of knowledge. In *CRYPTO*, pages 390–420, 1992. [1](#)
- [21] Mihir Bellare, Russell Impagliazzo, and Moni Naor. Does parallel repetition lower the error in computationally sound protocols? In *FOCS*, pages 374–383, 1997. [4.4.1](#)
- [22] Mihir Bellare and Adriana Palacio. The knowledge-of-exponent assumptions and 3-round zero-knowledge protocols. In *Proceedings of the 24th Annual International Cryptology Conference*, pages 273–289, 2004. [1](#), [4.1.3](#), [4.1.5](#), [4.1.6](#), [5.3.1.2](#)
- [23] Mihir Bellare and Adriana Palacio. Towards plaintext-aware public-key encryption without random oracles. In *ASIACRYPT*, pages 48–62, 2004. [4.1.3](#), [4.1.6](#), [1](#)
- [24] Eli Ben-Sasson, Alessandro Chiesa, Daniel Genkin, and Eran Tromer. Fast reductions from RAMs to delegatable succinct constraint satisfaction problems. In *Proceedings of the 4th Innovations in Theoretical Computer Science Conference, ITCS '13*, pages 401–414, 2013. [5.1.1](#), [5.1.4](#), [5.4.1.3](#), [120](#), [5.4.1.3](#)
- [25] Eli Ben-Sasson, Alessandro Chiesa, Eran Tromer, and Madars Virza. Scalable zero knowledge via cycles of elliptic curves. In *Proceedings of the 34th Annual International Cryptology Conference, CRYPTO '14*, 2014. [5.1.2](#)

- [26] Eli Ben-Sasson, Oded Goldreich, Prahladh Harsha, Madhu Sudan, and Salil Vadhan. Short PCPs verifiable in polylogarithmic time. In *Proceedings of the 20th Annual IEEE Conference on Computational Complexity*, CCC '05, pages 120–134, 2005. [5.3.2.3](#)
- [27] Siavosh Benabbas, Rosario Gennaro, and Yevgeniy Vahlis. Verifiable delegation of computation over large datasets. In *Proceedings of the 31st Annual International Cryptology Conference*, CRYPTO '11, pages 111–131, 2011. [5.1.1](#)
- [28] Nir Bitansky, Ran Canetti, Alessandro Chiesa, Shafi Goldwasser, Huijia Lin, Eran Tromer, and Aviad Rubinfeld. The haunting of the snark. *IACR Cryptology ePrint Archive*, 2014. [1](#), [1.2](#), [4.1.5](#), [4.1.5](#), [4.1.6](#), [5](#), [5.1.5](#)
- [29] Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, ITCS '12, pages 326–349, 2012. [1.2](#), [2](#), [63](#), [5.1.5](#), [100](#), [110](#)
- [30] Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. Recursive composition and bootstrapping for snarks and proof-carrying data. In *STOC*, pages 111–120, 2013. [1.2](#), [2](#), [5](#), [5.1.2](#), [93](#)
- [31] Nir Bitansky, Ran Canetti, Henry Cohn, Shafi Goldwasser, Yael Tauman Kalai, Omer Paneth, and Alon Rosen. The impossibility of obfuscation with auxiliary input or a universal simulator. *CoRR*, abs/1401.0348, 2014. [4.1.3](#)
- [32] Nir Bitansky, Ran Canetti, Omer Paneth, and Alon Rosen. On the existence of extractable one-way functions. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing*, STOC '14, 2014. [1.2](#), [4](#)
- [33] Nir Bitansky and Alessandro Chiesa. Succinct arguments from multi-prover interactive proofs and their efficiency benefits. In *CRYPTO*, pages 255–272, 2012. [2](#), [5.1.5](#)
- [34] Nir Bitansky, Alessandro Chiesa, Yuval Ishai, Rafail Ostrovsky, and Omer Paneth. Succinct non-interactive arguments via linear interactive proofs. In *TCC*, pages 315–333, 2013. [2](#), [5.1.1](#), [5.1.2](#), [5.1.2](#), [5.4](#)
- [35] Nir Bitansky and Omer Paneth. From the impossibility of obfuscation to a new non-black-box simulation technique. In *FOCS*, pages 223–232, 2012. [1.2](#), [3](#), [3.1.7](#)
- [36] Nir Bitansky and Omer Paneth. On the impossibility of approximate obfuscation and applications to resettable cryptography. In *STOC*, pages 241–250, 2013. [1.2](#), [3](#), [3.1.7](#)
- [37] Manuel Blum. Coin flipping by telephone. In *Proceedings of the 18th Annual International Cryptology Conference*, pages 11–15, 1981. [3.5.1](#)
- [38] Manuel Blum. How to prove a theorem so no one else can claim it. In *Proceedings of the International Congress of Mathematicians*, pages 1444–1451, 1986. [3.5.1](#), [3.5.1.1](#), [3.5.4.2](#), [4.5.2](#)

- [39] Manuel Blum, Will Evans, Peter Gemmell, Sampath Kannan, and Moni Naor. Checking the correctness of memories. In *Proceedings of the 32nd Annual Symposium on Foundations of Computer Science*, FOCS '91, pages 90–99, 1991. [5.1.4](#), [5.4.1.3](#), [120](#), [5.4.1.3](#)
- [40] Dan Boneh, Gil Segev, and Brent Waters. Targeted malleability: homomorphic encryption for restricted computations. In *ITCS*, pages 350–366, 2012. [2](#), [5.1.2](#)
- [41] Dan Boneh and Brent Waters. Constrained pseudorandom functions and their applications. In *ASIACRYPT (2)*, pages 280–300, 2013. [4.1.3](#), [4.3.2](#)
- [42] Dan Boneh and Mark Zhandry. Multiparty key exchange, efficient traitor tracing, and more from indistinguishability obfuscation. In *CRYPTO (1)*, pages 480–499, 2014. [4.1.3](#)
- [43] Ravi B. Boppana, Johan Håstad, and Stathis Zachos. Does co-NP have short interactive proofs? *Information Processing Letters*, 25(2):127–132, 1987. [5.1.1](#)
- [44] Elette Boyle, Shafi Goldwasser, and Ioana Ivan. Functional signatures and pseudorandom functions. In *Public Key Cryptography*, pages 501–519, 2014. [4.1.3](#), [4.3.2](#)
- [45] Elette Boyle and Rafael Pass. Limits of extractability assumptions with distributional auxiliary input. *IACR Cryptology ePrint Archive*, 2013:703, 2013. [4.1.6](#)
- [46] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. *TOCT*, 6(3):13, 2014. [3.3.5](#)
- [47] Zvika Brakerski and Guy N. Rothblum. Virtual black-box obfuscation for all circuits via generic graded encoding. In *TCC*, pages 1–25, 2014. [1](#), [4.3.1](#)
- [48] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) lwe. In *FOCS*, pages 97–106, 2011. [4.4.1](#), [5.3.2.2](#), [5.3.3.2](#), [108](#)
- [49] Gilles Brassard, David Chaum, and Claude Crépeau. Minimum disclosure proofs of knowledge. *Journal of Computer and System Sciences*, 37(2):156–189, 1988. [5.1.1](#)
- [50] Christian Cachin, Silvio Micali, and Markus Stadler. Computationally private information retrieval with polylogarithmic communication. In *Proceedings of the International Conference on the Theory and Application of Cryptographic Techniques*, ASIACRYPT '99, pages 402–414, 1999. [5.3.2.2](#), [5.3.2.2](#)
- [51] Ran Canetti and Ronny Ramzi Dakdouk. Extractable perfectly one-way functions. In *Proceedings of the 35th International Colloquium on Automata, Languages and Programming*, pages 449–460, 2008. [1](#), [4.1.5](#), [56](#)
- [52] Ran Canetti and Ronny Ramzi Dakdouk. Towards a theory of extractable functions. In *TCC*, pages 595–613, 2009. [1](#)
- [53] Ran Canetti, Oded Goldreich, Shafi Goldwasser, and Silvio Micali. Resettable zero-knowledge (extended abstract). In *STOC*, pages 235–244, 2000. [1.1](#), [3.1.5](#), [3.5.1](#), [40](#), [3.5.1](#), [3.5.1.1](#), [3.5.1.1](#)

- [54] Ran Canetti, Huijia Lin, and Omer Paneth. Public-coin concurrent zero-knowledge in the global hash model. In *TCC*, pages 80–99, 2013. [3.1.1](#), [3.1.1](#), [3.1.5](#)
- [55] Ran Canetti, Huijia Lin, and Rafael Pass. Adaptive hardness and composable security in the plain model from standard assumptions. In *FOCS*, pages 541–550, 2010. [3.5.4.3](#)
- [56] Ran Canetti, Ben Riva, and Guy N. Rothblum. Two 1-round protocols for delegation of computation. Cryptology ePrint Archive, Report 2011/518, 2011. [5.1.1](#)
- [57] Alessandro Chiesa and Eran Tromer. Proof-carrying data and hearsay arguments from signature cards. In *Proceedings of the 1st Symposium on Innovations in Computer Science*, ICS '10, pages 310–331, 2010. [5.1.2](#)
- [58] Chongwon Cho, Rafail Ostrovsky, Alessandra Scafuro, and Ivan Visconti. Simultaneously resettable arguments of knowledge. In *TCC*, pages 530–547, 2012. [3.1.1](#), [3](#), [3.1.5](#), [3.1.5](#), [3.1.6](#), [3.4.1](#), [3.5.6](#), [4.5.2](#)
- [59] Kai-Min Chung, Yael Kalai, and Salil Vadhan. Improved delegation of computation using fully homomorphic encryption. In *Proceedings of the 30th Annual International Cryptology Conference*, CRYPTO '10, pages 483–501, 2010. [5.1.1](#)
- [60] Kai-Min Chung, Rafail Ostrovsky, Rafael Pass, Muthuramakrishnan Venkitasubramaniam, and Ivan Visconti. 4-round resettable-sound zero knowledge. In *TCC*, pages 192–216, 2014. [3.1.5](#), [3.1.7](#)
- [61] Kai-Min Chung, Rafail Ostrovsky, Rafael Pass, and Ivan Visconti. Simultaneous resettable security from one-way functions. In *FOCS*, pages 60–69, 2013. [3.1.5](#), [3.1.7](#), [3.5.5](#), [50](#), [3.5.5](#)
- [62] Kai-Min Chung and Huijia Lin and Rafael Pass. Constant-round concurrent zero knowledge from p-certificates. In *FOCS*, 2013. [1](#)
- [63] Kai-Min Chung, Rafael Pass, and Karn Seth. Non-black-box simulation from one-way functions and applications to resettable security. In *STOC*, pages 231–240, 2013. [3.1.1](#), [2](#), [3.1.5](#), [3.1.5](#), [3.1.5](#), [3.1.7](#)
- [64] Graham Cormode, Michael Mitzenmacher, and Justin Thaler. Practical verified computation with streaming interactive proofs. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, ITCS '12, pages 90–112, 2012. [5.1.1](#)
- [65] Graham Cormode, Justin Thaler, and Ke Yi. Verifying computations with streaming interactive proofs. *Proceedings of the VLDB Endowment*, 5(1):25–36, 2011. [5.1.1](#)
- [66] Jean-Sébastien Coron, Tancrede Lepoint, and Mehdi Tibouchi. Practical multilinear maps over the integers. In *CRYPTO (1)*, pages 476–493, 2013. [1](#), [4.1.3](#)
- [67] Ivan Damgård. Towards practical public key systems secure against chosen ciphertext attacks. In *Proceedings of CRYPTO91*, pages 445–456, 1992. [1](#), [4.1.3](#), [4.1.6](#), [1](#), [5.1.5](#), [5.3.1.2](#)

- [68] Ivan Damgård, Sebastian Faust, and Carmit Hazay. Secure two-party computation with low communication. In *TCC*, pages 54–74, 2012. [2](#), [5.1.5](#), [109](#)
- [69] Yi Deng, Vipul Goyal, and Amit Sahai. Resolving the simultaneous resettability conjecture and a new non-black-box simulation strategy. In *FOCS*, pages 251–260, 2009. [3.1.1](#), [3.1.1](#), [2](#), [3.1.5](#), [3.1.5](#), [3.1.5](#), [3.1.7](#), [3.5.1](#), [3.5.4.3](#), [3.5.4.3](#), [3.5.5](#), [50](#), [3.5.5](#)
- [70] Alexander Dent and Steven Galbraith. Hidden pairings and trapdoor DDH groups. In Florian Hess, Sebastian Pauli, and Michael Pohst, editors, *Algorithmic Number Theory*, volume 4076 of *Lecture Notes in Computer Science*, pages 436–451. 2006. [4.1.6](#)
- [71] Giovanni Di Crescenzo and Helger Lipmaa. Succinct NP proofs from an extractability assumption. In *Proceedings of the 4th Conference on Computability in Europe*, pages 175–185, 2008. [1.1](#), [2](#), [5.1.1](#), [2](#), [5.1.1](#), [5.1.3](#), [5.1.3](#), [4](#), [5.1.3](#), [5](#), [5.3.3](#)
- [72] Yevgeniy Dodis, Roberto Oliveira, and Krzysztof Pietrzak. On the generic insecurity of the full domain hash. pages 449–466, 2005. [65](#)
- [73] Cynthia Dwork, Michael Langberg, Moni Naor, Kobbi Nissim, and Omer Reingold. Succinct NP proofs and spooky interactions, December 2004. Available at www.openu.ac.il/home/mikel/papers/spooky.ps. [5.1.1](#)
- [74] Cynthia Dwork and Moni Naor. Zaps and their applications. *SIAM J. Comput.*, 36(6):1513–1543, 2007. [3.1.4](#), [3.1.4](#), [3.1.6](#), [3.5.1](#), [2](#), [4.1.5](#)
- [75] Uriel Feige, Shafi Goldwasser, László Lovász, Shmuel Safra, and Mario Szegedy. Approximating clique is almost NP-complete (preliminary version). In *Proceedings of the 32nd Annual Symposium on Foundations of Computer Science*, SFCS '91, pages 2–12, 1991. [1.1](#)
- [76] Uriel Feige, Dror Lapidot, and Adi Shamir. Multiple noninteractive zero knowledge proofs under general assumptions. *SIAM J. Comput.*, 29(1):1–28, 1999. [3.1.3](#), [3.1.4](#), [4.1.5](#)
- [77] Uriel Feige and Adi Shamir. Zero knowledge proofs of knowledge in two rounds. In *CRYPTO*, pages 526–544, 1989. [1](#), [3.1.5](#), [3.5.1.1](#), [3.5.4.2](#)
- [78] Uriel Feige and Adi Shamir. Witness indistinguishable and witness hiding protocols. In *STOC*, pages 416–426, 1990. [3.1.3](#), [3.1.5](#), [3.1.6](#), [3.4.1](#), [3.5.6](#), [4.5.2](#)
- [79] Amos Fiat and Adi Shamir. How to prove yourself: practical solutions to identification and signature problems. In *Proceedings of the 6th Annual International Cryptology Conference*, CRYPTO '87, pages 186–194, 1987. [5.1.1](#)
- [80] Dario Fiore and Rosario Gennaro. Publicly verifiable delegation of large polynomials and matrix computations, with applications. Cryptology ePrint Archive, Report 2012/281, 2012. [5.1.1](#)
- [81] Sanjam Garg, Craig Gentry, and Shai Halevi. Candidate multilinear maps from ideal lattices. In *EUROCRYPT*, pages 1–17, 2013. [1](#), [4.1.3](#)

- [82] Sanjam Garg, Craig Gentry, Shai Halevi, and Mariana Raykova. Two-round secure mpc from indistinguishability obfuscation. In *TCC*, pages 74–94, 2014. [4.1.3](#)
- [83] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *FOCS*, 2013. [1](#), [4.1.2](#), [4.1.3](#), [4.3.1](#)
- [84] Rosario Gennaro, Craig Gentry, and Bryan Parno. Non-interactive verifiable computing: outsourcing computation to untrusted workers. In *Proceedings of the 30th Annual International Cryptology Conference, CRYPTO '10*, pages 465–482, 2010. [5.1.1](#)
- [85] Rosario Gennaro, Craig Gentry, Bryan Parno, and Mariana Raykova. Quadratic span programs and succinct NIZKs without PCPs. Cryptology ePrint Archive, Report 2012/215, 2012. [5.1.1](#), [5.1.2](#), [5.1.2](#), [5.4](#)
- [86] Rosario Gennaro, Craig Gentry, Bryan Parno, and Mariana Raykova. Quadratic span programs and succinct nizks without pcps. In *EUROCRYPT*, pages 626–645, 2013. [2](#)
- [87] Rosario Gennaro, Hugo Krawczyk, and Tal Rabin. Okamoto-Tanaka revisited: Fully authenticated Diffie-Hellman with minimal overhead. In *Proceedings of the 8th International Conference on Applied Cryptography and Network Security, ACNS '10*, pages 309–328, 2010. [4.1.6](#)
- [88] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC '09*, pages 169–178, 2009. [3.3.5](#), [5.1.4](#)
- [89] Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. *i*-hop homomorphic encryption and rerandomizable yao circuits. In *CRYPTO*, pages 155–172, 2010. [2](#), [4.1.5](#), [4.5.3](#)
- [90] Craig Gentry and Zulfikar Ramzan. Single-database private information retrieval with constant communication rate. In *Proceedings of the 32nd International Colloquium on Automata, Languages and Programming, ICALP '05*, pages 803–815, 2005. [5.1.5](#)
- [91] Craig Gentry and Daniel Wichs. Separating succinct non-interactive arguments from all falsifiable assumptions. In *Proceedings of the 43rd Annual ACM Symposium on Theory of Computing*, pages 99–108, 2011. [1](#), [1.1](#), [1](#), [5.1.1](#)
- [92] O. Goldreich and L. A. Levin. A hard-core predicate for all one-way functions. In *STOC '89: Proceedings of the twenty-first annual ACM symposium on Theory of computing*, pages 25–32, New York, NY, USA, 1989. ACM. [5](#), [3.2.2](#)
- [93] Oded Goldreich. *Foundations of Cryptography: Basic Tools*. Cambridge University Press, New York, NY, USA, 2000. [2](#), [3.3.1](#)
- [94] Oded Goldreich. *Foundations of Cryptography: Volume 2, Basic Applications*. Cambridge University Press, New York, NY, USA, 2004. [2](#), [4.1.3](#), [4.5.1](#)
- [95] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *J. ACM*, 33(4):792–807, 1986. [4](#), [4.1.3](#), [4.3.2](#)

- [96] Oded Goldreich and Johan Håstad. On the complexity of interactive proofs with bounded communication. *Information Processing Letters*, 67(4):205–214, 1998. [5.1.1](#)
- [97] Oded Goldreich and Ariel Kahan. How to construct constant-round zero-knowledge proof systems for np. *J. Cryptology*, 9(3):167–190, 1996. [3.5.4.2](#)
- [98] Oded Goldreich and Hugo Krawczyk. On the composition of zero-knowledge proof systems. *SIAM J. Comput.*, 25(1):169–192, 1996. [1](#), [1.1](#), [3.1.1](#), [3.1.1](#), [4.1.5](#), [4.6](#)
- [99] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game. In *STOC '87: Proceedings of the nineteenth annual ACM symposium on Theory of computing*, pages 218–229, 1987. [1](#)
- [100] Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity for all languages in np have zero-knowledge proof systems. *J. ACM*, 38(3):691–729, 1991. [3.1.1](#)
- [101] Oded Goldreich and Yair Oren. Definitions and properties of zero-knowledge proof systems. *Journal of Cryptology*, 7(1):1–32, December 1994. [4.1.5](#), [4.6](#)
- [102] Oded Goldreich, Salil Vadhan, and Avi Wigderson. On interactive proofs with a laconic prover. *Computational Complexity*, 11(1/2):1–53, 2002. [5.1.1](#)
- [103] Shafi Goldwasser and Yael Tauman Kalai. On the impossibility of obfuscation with auxiliary input. In *FOCS*, pages 553–562, 2005. [4.1.3](#)
- [104] Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. Delegating computation: Interactive proofs for Muggles. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing*, STOC '08, pages 113–122, 2008. [5.1.1](#)
- [105] Shafi Goldwasser, Huijia Lin, and Aviad Rubinfeld. Delegation of computation without rejection problem from designated verifier CS-proofs. Cryptology ePrint Archive, Report 2011/456, 2011. [2](#), [5.1.5](#)
- [106] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM J. Comput.*, 18(1):186–208, 1989. [1](#), [3.1.1](#), [4.5.1](#), [5.1.1](#)
- [107] Vipul Goyal. Non-black-box simulation in the fully concurrent setting. In *STOC*, pages 221–230, 2013. [3.1.1](#), [3.1.1](#), [3.1.7](#)
- [108] Vipul Goyal and Abhishek Jain. On the round complexity of covert computation. In *STOC*, pages 191–200, 2010.
- [109] Vipul Goyal and Hemanta K. Maji. Stateless cryptographic protocols. In *FOCS*, pages 678–687, 2011. [3.1.1](#)
- [110] Vipul Goyal and Amit Sahai. Resetably secure computation. In *EUROCRYPT*, pages 54–71, 2009. [3.1.1](#), [3.1.1](#)
- [111] Jens Groth. Short pairing-based non-interactive zero-knowledge arguments. In *Proceedings of the 16th International Conference on the Theory and Application of Cryptology and Information Security*, pages 321–340, 2010. [2](#), [5.1.1](#), [5.1.2](#), [5.1.2](#), [5.3.1.2](#), [5.4](#)

- [112] Divya Gupta and Amit Sahai. On constant-round concurrent zero-knowledge from a knowledge assumption. *IACR Cryptology ePrint Archive*, 2012:572, 2012. [4.1.6](#)
- [113] Satoshi Hada and Toshiaki Tanaka. On the existence of 3-round zero-knowledge protocols. In *Proceedings of the 18th Annual International Cryptology Conference*, pages 408–423, 1998. [1](#), [4.1.2](#), [4.1.3](#), [4.1.3](#), [4.1.5](#), [4.1.6](#), [5.3.1.2](#)
- [114] Iftach Haitner and Thomas Holenstein. On the (im)possibility of key dependent encryption. pages 202–219, 2009. [65](#)
- [115] Shai Halevi and Yael Tauman Kalai. Smooth projective hashing and two-message oblivious transfer. *J. Cryptology*, 25(1):158–193, 2012. [4.5.3](#)
- [116] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM J. Comput.*, 28(4):1364–1396, 1999. [4](#)
- [117] Susan Hohenberger, Amit Sahai, and Brent Waters. Replacing a random oracle: Full domain hash from indistinguishability obfuscation. In *EUROCRYPT*, pages 201–220, 2014. [4.1.3](#)
- [118] Yael Tauman Kalai and Ran Raz. Probabilistically checkable arguments. In *Proceedings of the 29th Annual International Cryptology Conference*, CCC '09, pages 143–159, 2009. [5.1.1](#)
- [119] Yael Tauman Kalai, Ran Raz, and Ron D. Rothblum. How to delegate computations: The power of no-signaling proofs. In *STOC*, 2014. [4.1.2](#), [4.1.4](#), [4.4](#), [4.4.1](#), [72](#), [75](#), [5.1.1](#)
- [120] Aggelos Kiayias, Stavros Papadopoulos, Nikos Triandopoulos, and Thomas Zacharias. Delegatable pseudorandom functions and applications. In *ACM Conference on Computer and Communications Security*, pages 669–684, 2013. [4.1.3](#), [4.3.2](#)
- [121] Joe Kilian. A note on efficient zero-knowledge proofs and arguments. In *Proceedings of the 24th Annual ACM Symposium on Theory of Computing*, pages 723–732, 1992. [5.1.1](#), [5.1.3](#)
- [122] Joe Kilian and Erez Petrank. Concurrent and resettable zero-knowledge in polynomial algorithm rounds. In *STOC*, pages 560–569, 2001. [3.1.5](#)
- [123] Dror Lapidot and Adi Shamir. Publicly verifiable non-interactive zero-knowledge proofs. In *CRYPTO*, pages 353–365, 1990. [4.5.2](#)
- [124] Helger Lipmaa. Progression-free sets and sublinear pairing-based non-interactive zero-knowledge arguments. In *TCC*, pages 169–189, 2012. [2](#), [5.4](#)
- [125] Helger Lipmaa. Progression-free sets and sublinear pairing-based non-interactive zero-knowledge arguments. In *Proceedings of the 9th Theory of Cryptography Conference*, TCC '12, pages 169–189, 2012. [5.1.1](#), [5.1.2](#), [5.1.2](#)
- [126] Helger Lipmaa. Succinct non-interactive zero knowledge arguments from span programs and linear error-correcting codes. In *ASIACRYPT (1)*, pages 41–60, 2013. [2](#)

- [127] Ralph C. Merkle. A certified digital signature. In *Proceedings of the 9th Annual International Cryptology Conference, CRYPTO '89*, pages 218–238, 1989. [5.1.3](#), [5.3.2.1](#)
- [128] Silvio Micali. Computationally sound proofs. *SIAM Journal on Computing*, 30(4):1253–1298, 2000. Preliminary version appeared in FOCS '94. [1.1](#), [4.1.2](#), [5.1.1](#), [5.1.1](#)
- [129] Silvio Micali and Leonid Reyzin. Min-round resettable zero-knowledge in the public-key model. In *EUROCRYPT*, pages 373–393, 2001. [3.1.1](#), [3.5.1](#)
- [130] Thilo Mie. Polylogarithmic two-round argument systems. *Journal of Mathematical Cryptology*, 2(4):343–363, 2008. [2](#), [5.1.5](#)
- [131] Ellis D. Miner, Randii R. Wessen, and Jeffrey N. Cuzzi. *The scientific significance of planetary ring systems*. Springer Praxis Books. 2007. [2](#)
- [132] Tal Moran and Alon Rosen. There is no indistinguishability obfuscation in pessiland. *IACR Cryptology ePrint Archive*, 2013:643, 2013. [4.3.3.3](#)
- [133] Moni Naor. Bit commitment using pseudorandomness. *J. Cryptology*, 4(2):151–158, 1991. [3.5.1](#), [3.5.5](#)
- [134] Moni Naor. On cryptographic assumptions and challenges. In *Proceedings of the 23rd Annual International Cryptology Conference*, pages 96–109, 2003. [1](#), [4.1.2](#), [65](#), [5.1.1](#)
- [135] Moni Naor and Benny Pinkas. Efficient oblivious transfer protocols. In *SODA*, pages 448–457, 2001. [4.5.3](#)
- [136] Eiji Okamoto and Kazue Tanaka. Key distribution system based on identification information. *Selected Areas in Communications, IEEE Journal on*, 7(4):481–485, May 1989. [4.1.6](#)
- [137] Rafail Ostrovsky and Ivan Visconti. Simultaneous resettability from collision resistance. *Electronic Colloquium on Computational Complexity (ECCC)*, 2012. [4.5.2](#)
- [138] Rafael Pass. Simulation in quasi-polynomial time, and its application to protocol composition. In *EUROCRYPT*, pages 160–176, 2003. [3.1.1](#)
- [139] Rafael Pass and Alon Rosen. Bounded-concurrent secure two-party computation in a constant number of rounds. In *FOCS*, pages 404–413, 2003.
- [140] Rafael Pass and Alon Rosen. New and improved constructions of nonmalleable cryptographic protocols. *SIAM J. Comput.*, 38(2):702–752, 2008.
- [141] Rafael Pass, Alon Rosen, and Wei-Lung Dustin Tseng. Public-coin parallel zero-knowledge for np. 2011. [3.1.1](#)
- [142] Rafael Pass, Wei-Lung Dustin Tseng, and Douglas Wikström. On the composition of public-coin zero-knowledge protocols. *SIAM J. Comput.*, 40(6):1529–1553, 2011. [3.1.1](#), [3.5.1](#)
- [143] Rafael Pass and Hoeteck Wee. Black-box constructions of two-party protocols from one-way functions. In *TCC*, pages 403–418, 2009. [4.1.5](#), [4.5.2](#)

- [144] Manoj Prabhakaran, Alon Rosen, and Amit Sahai. Concurrent zero knowledge with logarithmic round-complexity. In *FOCS*, pages 366–375, 2002. [3.1.5](#)
- [145] Manoj Prabhakaran and Rui Xue. Statistically hiding sets. In *Proceedings of the The Cryptographers’ Track at the RSA Conference 2009*, CT-RSA 2009, pages 100–116, 2009. [4.1.6](#)
- [146] Ransom Richardson and Joe Kilian. On the concurrent composition of zero-knowledge proofs. In *EUROCRYPT*, pages 415–431, 1999. [3.1.1](#), [3.1.5](#), [3.5.4.3](#)
- [147] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In *STOC*, pages 475–484, 2014. [4.1.3](#), [4.1.3](#), [4.3.2](#)
- [148] Adi Shamir. $IP = PSPACE$. *Journal of the ACM*, 39(4):869–877, 1992. [5.1.1](#)
- [149] Paul Valiant. Incrementally verifiable computation or proofs of knowledge imply time/space efficiency. In *Proceedings of the 5th Theory of Cryptography Conference*, TCC ’08, pages 1–18, 2008. [5.1.2](#), [5.1.4](#), [5.3.2.3](#)
- [150] Hoeteck Wee. On round-efficient argument systems. In *Proceedings of the 32nd International Colloquium on Automata, Languages and Programming*, ICALP ’05, pages 140–152, 2005. [5.1.1](#)
- [151] Andrew Chi-Chih Yao. Theory and applications of trapdoor functions (extended abstract). In *23rd Annual Symposium on Foundations of Computer Science, Chicago, Illinois, USA, 3-5 November 1982*, pages 80–91, 1982. [3.1.6](#), [3.2.1](#)
- [152] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *FOCS*, pages 162–167, 1986. [4.5.3](#)