

NON-BLACK BOX USE OF CODE IN CRYPTOGRAPHY

A Dissertation

Presented to the Faculty of the Graduate School
of Cornell University

in Partial Fulfillment of the Requirements for the Degree of
Doctor of Philosophy

by

Karn Seth

February 2016

© 2016 Karn Seth
ALL RIGHTS RESERVED

NON-BLACK BOX USE OF CODE IN CRYPTOGRAPHY

Karn Seth, Ph.D.

Cornell University 2016

In this thesis, I broadly explore how we can incorporate program code in cryptography, both in terms of improving existing cryptographic primitives, and in constructing and analyzing completely new primitives.

Most classical cryptography models adversaries as a “black-box”, meaning that the security argument only uses the input-output behavior of the adversaries, without using their actual *code*. I show, along the lines of [7], how to use “non-black-box” techniques, using the program code of adversaries to improve constructions of resettably-sound zero-knowledge protocols (RSZK), and to reduce the required security assumption to the minimum possible (one-way-functions). I also show how the code of adversaries can be used more directly, to construct what we believe is the first practical candidate RSZK protocol.

In a very different way of leveraging code in cryptography, I also investigate several new cryptographic primitives: indistinguishability obfuscators (IO), randomized encoding (RE), and functional encryption (FE). These primitives inherently leverage program code to provide new functionality beyond that offered by classical primitives, and they have allowed novel applications like delegatable computation. In particular, I investigate the security of multilinear maps, the crucial building block for many of these new primitives, and give the first construction of IO that has a security reduction to a concrete hardness assumption on multilinear maps.

I also consider the interrelations between these new primitives. In particular,

I investigate how “compact” RE can be used to build IO, a qualitatively stronger primitive. Using very different techniques, I also study how a very weak and inefficient version of IO can be bootstrapped to “standard” IO, by using FE as an intermediate primitive.

BIOGRAPHICAL SKETCH

Karn spent four years earning his B.A. at Dartmouth College, under the watchful eye of the Faculty of Computer Science and Mathematics. After graduating in 2010, the granite of New Hampshire having been sufficiently installed in his brain, he moved from wintry Hanover to winterier Ithaca, for his PhD in computer science at Cornell University. His research was in the area of theoretical cryptography, where he was advised by Professor Rafael Pass. After two wonderful years in Ithaca, he followed Rafael to the new Cornell Tech campus in New York City, where he completed the rest of his PhD work. Along the way, he had two productive internships at Google, where he worked on things he can't talk about.

Hi Mom!
This thesis is dedicated to you.

ACKNOWLEDGEMENTS

Deepest thanks to my advisor, Rafael Pass, and to my collaborators Rachel Lin, Kai-Min Chung and Sidharth Telang, for all their help over the course of my PhD. I learned everything I know about cryptography from working with them.

Thanks also to my committee members Ari Juels and Eva Tardos for their patient contributions to this PhD process. Also, thanks to Moti Yung and Sarvar Patel for a pair of fun and inspiring internships at Google. Also, big thanks to Sid, Stavros, Adith, Z, Brian and Eric, for making this a fun 5 years in Ithaca and New York City.

Finally, thanks to you, the reader, for making it this far, and for checking out what I spent the last few years working on!

TABLE OF CONTENTS

Biographical Sketch	iii
Dedication	iv
Acknowledgements	v
Table of Contents	vi
1 Introduction	1
1.1 Using Non-Black-Box Techniques in Classical Cryptography	4
1.2 Leveraging Program Code to Construct New Primitives	6
1.2.1 Investigating the Security of Multilinear Maps	8
1.2.2 Building IO from Compact Randomized Encodings	10
1.2.3 Bootstrapping Weak IO	12
2 Non-Black-Box Simulation from One-Way Functions And Applications to Resettable Security	14
2.1 Introduction	14
2.1.1 Our Result	17
2.1.2 Our Techniques	18
2.1.3 Subsequent Work	25
2.1.4 Outline	26
2.2 Preliminaries	27
2.2.1 Notations	27
2.2.2 Computational Indistinguishability	28
2.2.3 Interactive Arguments	29
2.2.4 Witness Indistinguishability	30
2.2.5 Commitment Schemes	31
2.2.6 Zero Knowledge	32
2.2.7 Resettable Sound Zero Knowledge	32
2.3 Signature Trees	35
2.3.1 Sig-Com Schemes	39
2.4 Oracle-Aided Resettable-sound Zero Knowledge Protocols	42
2.4.1 Oracle-aided Universal Arguments	45
2.4.2 Oracle-aided Zero-Knowledge Protocols	51
2.5 Resettable-sound Zero Knowledge in the Plain Model	59
2.6 PCP-Free RSZK	64
2.6.1 Proof of Argument of Knowledge property	71
2.6.2 Proof of Zero Knowledge property	91
2.7 Applications	97
3 Indistinguishability Obfuscation from Semantically-Secure Multilinear Encodings	101
3.1 Introduction	101
3.1.1 Towards “Provably-Secure” Obfuscation	104

3.1.2	Security of Multilinear (Graded) Encodings	106
3.1.3	Alternative Security Notions for Multilinear Encodings . .	114
3.1.4	Construction Overview	115
3.1.5	Discussion and Future Work	123
3.1.6	Outline of the Paper	127
3.2	Preliminaries	127
3.2.1	Obfuscation	129
3.2.2	Branching programs for NC^1	131
3.3	Semantically Secure Graded Encoding Schemes	133
3.3.1	Graded Encoding Schemes	133
3.3.2	Semantical Security	136
3.4	iO from Semantically Secure Multilinear Encodings	143
3.4.1	Neighboring-Matrix Indistinguishability Obfuscation ($nm-iO$)	144
3.4.2	From $nm-iO$ to iO	144
3.4.3	From Semantic Security to $nm-iO$	156
3.4.4	Achieving Obfuscation for Arbitrary Programs	180
3.5	iO from Single-Distribution Semantical Security	181
3.5.1	Single-Distribution Semantical Security	182
3.5.2	Basing Security on Single-Distribution Semantical Security	184
3.6	Alternative Security Notions of Semantical Security Encodings . .	189
3.6.1	Semantical Security w.r.t. Algebraic Attackers	190
3.6.2	Uber Assumptions for Multilinear Encodings	192
3.6.3	Strong Semantical and Uber Security	200
3.6.4	Weak Semantic Security	201
3.7	Technical Lemma	202
3.8	Proof of Lemma 33	203
3.9	Proof of Lemma 21	208
3.10	Acknowledgments	210
4	Output-Compressing Randomized Encodings and Applications	212
4.1	Introduction	212
4.1.1	Our results	216
4.2	Preliminaries	224
4.2.1	Concrete Security	224
4.2.2	Standard cryptographic primitives	226
4.2.3	Indistinguishability Obfuscation	226
4.2.4	Functional Encryption	229
4.3	Randomized Encoding Schemes	231
4.3.1	Randomized Encoding with Simulation Security	232
4.3.2	Distributional Indistinguishability Security	234
4.3.3	Compactness and Sublinear Compactness	236
4.3.4	Composition of Ind-Security	237
4.3.5	Simulation Security implies Indistinguishability Security .	242

4.4	Unbounded-Input IO from Compact RE	244
4.4.1	Security Proof	248
4.4.2	Nice Distributions	254
4.5	Bounded-Input IO from Sublinear RE	255
4.6	Bounded-Input IO from Compact RE in the CRS Model	262
4.6.1	Randomized Encoding Schemes in the CRS model	262
4.6.2	Succinctness and Weak Sublinear Compactness	265
4.6.3	Randomized encodings with CRS from Compact Functional Encryption	269
4.6.4	IO for Circuits from RE in the CRS model	275
4.6.5	Summary of Results using RE in the CRS model	279
4.7	Impossibility of Compact RE	279
5	Indistinguishability Obfuscation with Exponential Efficiency	287
5.1	Introduction	287
5.2	Preliminaries	293
5.2.1	Puncturable PRF	293
5.2.2	Functional Encryption	294
5.2.3	Indistinguishability Obfuscation	297
5.3	Exponentially-Efficient iO (\mathbf{XiO})	298
5.4	iO from \mathbf{XiO}	301
5.4.1	Weakly Sublinear Compact FE from Succinct FE and \mathbf{XiO} .	301
5.5	Comparison with [3]	306
	Bibliography	308

CHAPTER 1

INTRODUCTION

Classical cryptography deals with a vast number of different primitives and protocols, including encryption, digital signatures, commitment schemes, multi-party computation protocols (MPC), and zero knowledge protocols (ZK). However, many of the security proofs of these primitives model adversaries only as a “black-box”. To see what this means, consider the classic structure of a cryptographic security argument. In such arguments, the crucial step is usually to show that if an adversary \mathcal{A} breaks the security of a cryptographic primitive \mathcal{P} , then \mathcal{A} can be used to break some hard mathematical assumption \mathcal{M} underlying the construction of \mathcal{P} . This step is usually formalized by describing a reduction \mathcal{R} , that makes use of the adversary \mathcal{A} in order to break \mathcal{M} . A reduction \mathcal{R} is called “black-box” if the only way in which it uses \mathcal{A} is by invoking it on some number of inputs, and using its outputs, that is, if it only uses \mathcal{A} ’s input-output behavior as a black-box when breaking \mathcal{M} . In contrast, a reduction is called “non-black-box” if it accesses the explicit *program code* of \mathcal{A} in order to break \mathcal{M} .

Most cryptographic proofs of security, either implicitly or explicitly, limit themselves to black-box reductions. While this approach has achieved a great deal of success in constructing classical cryptographic primitives, a natural question is whether we can do better using non-black-box techniques. This leads us to the main question broadly examined in this thesis:

“How can we leverage the non-black-box use of program code in cryptography?”

I approach this question in two different ways. The first is by studying whether non-black-box techniques can allow us to improve results in *classical* cryptography, either in terms of efficiency, or by allowing us to weaken the required mathematical assumptions.

“Question 1: How can we use non-black-box techniques to improve constructions of classical cryptographic primitives?”

In answer to this question, I achieve the following:

- I construct the first protocol for a cryptographic primitive (resettably-sound zero knowledge) that can be proven secure from *minimal* assumptions
- I also show how the code of the adversary can be used in a much more direct way, by building a new protocol for resettably-sound zero knowledge, which we believe is the first potentially practical construction of the protocol.

In a very different way of using program code in cryptography, I also study several *new* cryptographic primitives: indistinguishability obfuscators (IO), randomized encoding (RE), and functional encryption (FE). These primitives inherently leverage program code to provide functionality beyond that offered by classical primitives, and have enabled a vast number of novel applications, a prominent example of which is delegatable computation. Indistinguishability obfuscators aim to scramble the code of a program, to hide internal keys and implementation details. Randomized encodings are a tool to allow delayed evaluation of a program Π on a single, pre-selected input x , while hiding both

Π and x . Functional encryption modifies traditional encryption to include so-called functional secret keys, that allow holders to decrypt a *function* of the encrypted message, but “nothing extra” about the message itself. Broadly, we ask the question:

“Question 2: How can we leverage program code to develop and analyze entirely new cryptographic primitives?”

To this end, I investigate the security underlying new constructions of the aforementioned primitives. I also study their interrelations: how one primitive can be used to build another. More specifically, I tackle the following questions:

- I investigate the security of multilinear maps, the crucial building block for many of these new primitives, and give the first construction of IO that has a security reduction to a concrete hardness assumption on multilinear maps.
- I investigate how “compact” RE can be used to build IO, a qualitatively stronger primitive.
- Using very different techniques, I also study how a very weak and inefficient version of IO can be bootstrapped to “standard” IO, by using FE as an intermediate primitive.

These two broad classes of questions, namely using non-black-box technique to improve classical techniques, and also to build and analyze completely new primitives, are explored in more detail below.

1.1 Using Non-Black-Box Techniques in Classical Cryptography

The first question we consider is whether we can use non-black-box techniques to improve classical primitives. To this end, Barak [7] had a breakthrough result showing for the first time that non-black box techniques can be used to overcome an impossibility result for a particular cryptographic primitive, public-coin zero knowledge. Zero-knowledge (ZK) protocols are two-party interactive protocols between a prover P and a verifier V , where P tries to convince V of the truth of some statement (in the canonical case, that a string x lies in some language L), without leaking “anything extra”. Public-coin ZK adds the further restriction that the verifier must output only random strings in each of its messages during the protocol. [45, 68] showed that if a public-coin ZK protocol has a black-box security reduction proving that the verifier learns “nothing extra”, then the same technique can be used by a malicious prover P^* to convince an honest verifier of a false statement

In a seminal work [7, 8], Barak gave the first construction of a public-coin ZK protocol, crucially using the *code* of the verifier in the security reduction. In doing so, he explicitly demonstrated that non-black-box use of code can overcome barriers that hold for black-box techniques. Subsequent work by [11] showed that these techniques could be used to construct an even stronger version of ZK, dubbed resettably-sound ZK. Resettably-sound ZK considers the setting when the verifier V is on an embedded device, and P is allowed to “reset” the device (e.g. by unplugging it and plugging it in again). As in the case of public-coin ZK, [11] show it is impossible to have RSZK with black-box security proofs. How-

ever, they also give a candidate protocol for RSZK, using Barak’s non-black-box techniques. Since these results, non-black-box security reductions have found applications in various other contexts (see e.g. [10, 104, 105, 52]).

One important limitation of the non-black-box reduction technique of Barak [7] (also present in its follow-up works) is that the technique requires stronger assumptions than those typically needed for constructing zero-knowledge protocols. In particular, the protocol of Barak (using the refinement in [10]) relies on the existence of families of collision-resistant hash functions (CRH), and as a consequence, such hash functions are needed in the above applications too. In contrast, for “plain” zero-knowledge (i.e., without, for instance, resettable soundness or the public-coin requirement) one-way functions (OWFs) are both sufficient and (essentially) necessary [69, 80, 103], leaving open the following question:

Do one-way functions suffice for performing non-black-box security reductions (for primitives that cannot be proven secure using black-box reductions)?

In Chapter 2, I show how, in a joint work with Rafael Pass and Kai-Min Chung, we can answer this question in the positive. Namely, we construct an RSZK protocol whose security can be based on only the existence of OWFs. At a high level, our approach is to replace the use of CRHs with a weaker primitive, digital signatures, that can be constructed using only OWFs, but which requires some extra interaction in order to preserve an equivalent level of security. We crucially modify the protocol along the lines of [105] so that the majority of this extra interaction only happens in the security reduction, but not in the “real”

execution of the protocol. This allows us to almost completely preserve both the security and the efficiency of the scheme of [7], while relying on a different assumption.

Also in Chapter 2, we give an entirely different approach for constructing RSZK protocols, which we believe is much more practical than Barak’s approach. At a very high level, a central sequence of steps in Barak’s protocol is to evaluate the verifier’s code on some particular input, record the (long) computation trace, produce a (very long) PCP π proving the correctness of this computation, and finally, prove knowledge of π . Importantly, even though the honest prover can use a “dummy” PCP π , the proof of knowledge of even this “dummy” is the same as that for a real PCP, which has a huge cost in practical terms. Our approach is to replace the “one-shot” evaluation of the entire verifier’s computation with a short interactive “slot” that allows provably executing just *one-step* of the verifier’s code. While the security proof must invoke this “slot” many times, the honest prover only needs to execute this slot once, with dummy values. As such, our changed protocol needs the prover to perform a dummy proof of only a single step of computation, instead of a dummy proof of knowledge of a large PCP for an entire computation. We thus believe our modified protocol is much more practical. Full details of the improved protocol are in Section 2.6.

1.2 Leveraging Program Code to Construct New Primitives

I consider three primitives, indistinguishability obfuscators, randomized encodings and functional encryption, that each inherently make use of program code

in order to provide new functionality.

Indistinguishability obfuscation are a type of program obfuscation, where the goal is to “scramble” a computer program, hiding its implementation details (making it hard to “reverse-engineer”), while preserving the functionality (i.e, input/output behavior) of the program. In recent years, the notion of *indistinguishability obfuscation (IO)* [12, 57] has emerged as the central notion of obfuscation: Roughly speaking, this notion requires that obfuscations $iO(C_1)$, $iO(C_2)$ of any two *functionally equivalent* circuits C_1 and C_2 (i.e., whose outputs agree on all inputs) from some class C (of circuits of some bounded size) are computationally indistinguishable. This notion of obfuscation has been found to have a plethora of amazing applications (see e.g., [116, 32, 30, 55, 19, 43, 90]).

Randomized encodings (RE), introduced by Ishai and Kushilevitz [82], aim to trade the computation of a “complex” (deterministic) function Π on a given input x for the computation of a “simpler” randomized function—the “encoding algorithm”—whose output distribution $\hat{\Pi}(x)$ encodes $\Pi(x)$ (from which $\Pi(x)$ can be efficiently decoded, or “evaluated”). Furthermore, the encoding $\hat{\Pi}(x)$ should not reveal anything beyond $\Pi(x)$. Randomized encodings, interesting in their own right, have recently become central in new constructions of indistinguishability obfuscators [90, 19].

In a Functional Encryption (FE) scheme [27], the owner of a master secret key can produce functional keys sk_f for functions f (usually represented as circuits or Turing Machines). Given an encryption of an input x computed using the public key pk and the functional key sk_f , anyone can compute $f(x)$, but nothing more about x itself. As such, functional encryption allows restricted decryption of ciphertexts, and has both direct applications, and has found use

in construction of other primitives, including IO [22].

In this thesis, I investigate the security of multilinear maps, the crucial building block for many of these new primitives, and give the first construction of IO that has a security reduction to a concrete hardness assumption on multilinear maps.

I also consider the interrelations between these new primitives. In particular, I investigate how “compact” RE can be used to build IO, a qualitatively stronger primitive. Using very different techniques, I also show how a very weak and inefficient version of IO can be bootstrapped to “standard” IO, by using FE as an intermediate primitive.

Each of these results is outlined in more detail below.

1.2.1 Investigating the Security of Multilinear Maps

In a breakthrough result, Garg, Gentry, Halevi, Raykova, Sahai, and Waters [57] provided the first candidate constructions of indistinguishability obfuscators for all polynomial-size circuits, based on so-called *multilinear maps* [28, 115, 54]—for which candidate constructions were put forward in a closely preceding seminal work of Garg, Gentry and Halevi [54]. However, the construction of IO in this work was essentially assumed to be secure, and did not give a security reduction to a concrete hardness assumption on multilinear maps. This left open the question:

Can the security of a general-purpose indistinguishability obfuscator be reduced to some “natural” intractability assumption on multilinear maps?

In joint work with Rafael Pass and Sidharth Telang, described in Chapter 3, we propose exactly such an intractability assumption on multilinear maps, which we call “semantic security”. At a very high level, multilinear maps allow messages to be encoded, and allow a set of operations: encodings can be added and/or multiplied to produce new encodings (subject to restrictions), and further, certain encodings can be “zero-tested”, to see if the value they contain is zero or non-zero. Intuitively, semantic security of multilinear maps implies that encodings of elements leak “nothing more” than can be learned through the legal application of these restricted operations. More formally, semantic security says that given any distribution that outputs two sets of messages M_1 and M_2 and a set of auxiliary messages Z such that the output of any legal sequence of operations on either (M_1, Z) or (M_2, Z) is the same (or statistically close), then the encodings $(\text{Enc}(M_1), \text{Enc}(Z))$ and $(\text{Enc}(M_2), \text{Enc}(Z))$ using the multilinear map are computationally indistinguishable.

We explore several variants of the semantic security assumption, and show that one particularly strongest version, though natural at first, is too strong, and is ruled out by the impossibility result of [13]. However, the more carefully weakened versions, including the very weakest version dubbed “entropic semantic security” (see Section 3.3.2), are still sufficient to build indistinguishability obfuscators. To prove this latter statement, we give a new construction of indistinguishability obfuscators whose security proof can be broken up into a series of short steps to which semantic security can be applied. More details of the construction can be found in Section 3.4.

Several works after ours have proposed different security assumptions, most prominently the Multilinear Subgroup Elimination Assumption [64] on

composite-order multilinear maps, a somewhat more powerful variant of multilinear maps. However, this assumption was later found to be broken in a sequence of attacks on multilinear maps [62, 47]. This sequence of attacks was followed by some attempted fixes [61, 29], which were also found to be broken [50]. After these attacks, the current state of multilinear map security is quite cloudy, but it appears that our notion of “entropic semantic security” still remains unbroken.

1.2.2 Building IO from Compact Randomized Encodings

Randomized encodings have recently been the subject of a great deal of research attention, and have proven to be extremely closely linked to obfuscators, especially in the setting of Turing Machines. [19] recently initiated a study of *succinct randomized encodings* for Turing Machines, where we require that the *time* required to compute an encoding $\hat{\Pi}(x)$ is smaller than the time required to compute $\Pi(x)$; their study focused on functions Π that have *single-bit* outputs. [19, 43, 90] show that subexponentially-secure indistinguishability obfuscators (*iO*) and one-way functions imply the existence of such succinct randomized encodings for all polynomial-time Turing machines that output just a single bit.

In Chapter 4, I present a joint work with Rafael Pass, Huijia Lin and Sidharth Telang, where further the study of such objects, focusing on functions Π with *long* outputs. That is, we consider a notion of *compact* randomized encodings, where the time to encode $\hat{\Pi}(x)$ is independent of both the running time of the machine, and the length of the output $\Pi(x)$. We show first that such randomized encodings cannot satisfy the usual notion of simulation based security, and

thus propose an alternative indistinguishability-based notion of security. What this roughly says is that if we have a distribution over pairs of machines and messages (Π_1, x_1, Π_2, x_2) such that the distributions of outputs $\Pi_1(x_1)$ and $\Pi_2(x_2)$ are (subexponentially) indistinguishable, then the encodings $\hat{\Pi}_1(x_1)$ and $\hat{\Pi}_2(x_2)$ are also (subexponentially) indistinguishable.

We show in Section 4.4 that compact RE is quite powerful, and that if it exists, it can be used to build IO for *unbounded-input* Turing Machines. This is surprising because RE can be viewed as a “degenerate” version of IO, that only works on one input. Our construction thus builds a seemingly stronger primitive out of a weaker one. In fact, we show that one can build IO out of a weaker notion of compact RE, which we call “sublinear” RE (though the resulting IO only works for bounded-input TMs).

However, these results turn out to be too good to be true, because they lead to a contradiction: we show in Section 4.7 that if compact RE exists, then it can be used to build distributions of machines that break indistinguishability security. This negative result is especially interesting because it crucially relies on our previously mentioned positive result building IO from compact RE.

However, we show that our results can be salvaged if we restrict ourselves to the Common Reference String (CRS) model. In this model, the encoding and decoding procedures are allowed to access a CRS that has been generated during a trusted setup phase. We show that, in this setting, compact RE with indistinguishability security is still possible, and we give a construction of it from compact functional encryption. We also show that our previous result building IO from compact RE carries over to the CRS model, with the caveat that the IO must be bounded-input. Finally, we show again that sublinear RE suffices for

this construction, and that sublinear RE can be constructed from sublinear FE. In doing so, we also show that sublinear FE implies IO for bounded-input Turing Machines, thus achieving the same result as [22, 3], albeit in a very different way.

1.2.3 Bootstrapping Weak IO

Despite amazing progress, to date, all candidate constructions of *IO* rely on candidate constructions of *multi-linear maps* [54, 49, 61, 51], all of which have non-trivial attacks [47, 97], and it is not clear to what extent the security of the obfuscators that rely on them are affected.

In Chapter 5, I present a joint work with Rafael Pass, Sidharth Telang and Huijia Lin, where rather than studying new candidate constructions of *IO*, we focus on identifying *weaker* notions of indistinguishability obfuscation that can amplified/bootstrapped into the standard notion of *iO*. We identify one notion, which we term *Exponentially Efficient IO (XIO)*, and show that, assuming the hardness of the Learning With Errors problem, XIO can be bootstrapped into “standard” IO for all polynomial sized circuits.

To understand the definition of XIO, first consider that there exists a trivial procedure to compute an indistinguishability obfuscation: canonicalizing the circuit. Since two equivalent circuits have identical canonicalizations, this is a valid, information-theoretically secure IO process. However, canonicalizing a circuit C (for example, by converting it into a function table) is an inefficient process, and can take time polynomial $|C|$ and exponential in the input length n . We define XIO to be any indistinguishability obfuscator that is just slightly

better than this brute force approach. More specifically, we allow the obfuscation to run in time $\text{poly}(\lambda, |C|) * 2^n$, but require the size of the obfuscation to be $\text{poly}(\lambda, |C|) * 2^{n(1-\epsilon)}$, where λ is the security parameter and $\epsilon > 0$.

We show that even this very weak notion of obfuscation can be used to achieve standard IO (that runs in time $\text{poly}(\lambda, |C|)$). The high-level outline of our construction is as follows: On the one hand, assuming LWE, [73] show how to construct *succinct* functional encryption for circuits with 1-bit outputs. On the other hand, our work [92] given in Chapter 4 shows that *sublinear* functional encryption for circuits with multi-bit outputs is sufficient to construct IO for all polynomial-sized circuits. Thus it is sufficient to go from succinct functional encryption for circuits with 1-bit outputs to sublinear functional encryption for circuits with multi-bit outputs. Unfortunately, the natural technique to modify succinct FE for circuits with 1-bit outputs to handle circuits with multi-bit outputs does not work because the resulting scheme has long ciphertexts, in particular, too long for the resulting FE scheme to be sublinear. Our solution, following an idea of [3], is to compress the ciphertext by releasing a circuit $G[K, m]$, that on input i , outputs the i^{th} piece of $\text{Enc}(m)$. We show that the slight compression afforded by XIO is exactly enough to make the resulting scheme a sublinear FE scheme for circuits with multi-bit outputs. Thus, by showing that XIO can be used to convert succinct FE to sublinear FE, we show that XIO together with the hardness of LWE is sufficient to achieve IO with standard efficiency for all polynomial-sized circuits.

CHAPTER 2

NON-BLACK-BOX SIMULATION FROM ONE-WAY FUNCTIONS AND APPLICATIONS TO RESETTABLE SECURITY

This section contains joint work with Kai-Min Chung (Academia Sinica) and Rafael Pass (Cornell University). It appeared in the proceedings of the Symposium of Theory of Computing (STOC) 2013.

2.1 Introduction

Zero-knowledge (\mathcal{ZK}) interactive protocols [76] are paradoxical constructs that allow one player (called the Prover) to convince another player (called the Verifier) of the validity of a mathematical statement $x \in L$, while providing *zero additional knowledge* to the Verifier. Beyond being fascinating in their own right, \mathcal{ZK} proofs have numerous cryptographic applications and are one of the most fundamental cryptographic building blocks.

The zero-knowledge property is formalized using the so-called *simulation paradigm*: for every malicious verifier V^* , we require the existence of a “simulator” S that, given just the input x , can indistinguishably reproduce the view of V^* in an interaction with the honest prover. (We note that the simulation paradigm extends well beyond the notion of zero-knowledge, and is a crucial component of modern definitions of protocol security.) The most typical way of performing such a simulation is using *black-box simulation* [70]: here we exhibit a universal simulator S that, given only black-box access to *any* (efficient) V^* , can reproduce the view of V^* in an interaction with the honest prover. Indeed most zero-knowledge protocols (and more generally protocols for secure com-

putation) are analyzed using black-box simulators. But several limitations of black-box simulators are also known; see e.g. [68, 45, 11, 110].

In a breakthrough result from 2001, Barak [7] demonstrated a new, powerful *non-black-box* simulation technique, and used this technique to construct a constant-round *public-coin* zero-knowledge argument; by the result of [68] such protocols cannot be proved zero-knowledge using just black-box simulation. In the same year, Barak, Goldwasser, Goldreich and Lindell [11] demonstrated that this non-black-box simulation technique could be used to achieve a *new cryptographic primitive* that cannot be proven secure using black-box simulation, namely *resettably-sound zero-knowledge protocols*. In a resettably-sound zero-knowledge protocol, the soundness property is required to hold even if the malicious prover is allowed to “reset” and “restart” the verifier. This model is particularly relevant for cryptographic protocols being executed on embedded devices, such as smart cards. (Since these devices have neither a built-in power supply, nor a non-volatile rewritable memory, they can be “reset” by simply disconnecting and reconnecting the power supply.) Roughly speaking, the reason why non-black-simulation is crucial for resettably-sound zero-knowledge protocols is that a black-box simulator has essentially the same “powers” as a malicious resetting prover (i.e., it can only reset and restart the verifier); from this observation it follows that, unless $L \in \text{BPP}$, a “good” simulator can be as a successful cheating prover. Since these results, non-black-box simulation techniques have found applications in various other contexts (see e.g. [10, 104, 105, 52]).

One important limitation of the non-black-box simulation technique of Barak [7] (also present in its follow-up works) is that the technique requires stronger assumptions than those typically needed for constructing zero-knowledge pro-

protocols. In particular, the protocol of Barak (using the refinement in [10]) relies on the existence of families of collision-resistant hash functions (CRH), and as a consequence, such hash functions are needed in the above applications too.¹ In contrast, for “plain” zero-knowledge (i.e., without, for instance, resettable soundness) one-way functions are both sufficient and (essentially) necessary [69, 80, 103], leaving open the following question, which is the focus of this work.

Do one-way functions suffice for performing non-black-box simulation (for primitives that cannot be proven secure using black-box simulation techniques)?

A very recent elegant work by Bitansky and Paneth [20] takes us a step closer to answering this question. They present a resettable-sound zero-knowledge argument without relying on hash functions; instead, they rely on the existence of an *oblivious transfer (OT) protocol*. Although, the existence of an OT protocol is seemingly a more “complex” assumption than the existence of CRHs,² it is not known whether the existence of an OT protocol implies the existence of CRH (or vice versa). More important, to achieve this result, Bitansky and Paneth devise a quite different method for performing non-black-box simulation.

¹The original protocol of Barak relies on a very slightly super-polynomially hard collision-resistant hash function; the need for super-polynomial hardness was removed in [10].

²Most candidate constructions of OT protocols rely on “structured”, number-theoretic or lattice-based, assumptions. Additionally, all these assumptions are known to imply also the existence of collision-resistant hash function (but the converse is not true).

2.1.1 Our Result

In this work, we answer the above question in the affirmative. We show that for the case of resettably-sound zero-knowledge, the existence of one-way functions suffices.

Theorem 1 (Main Theorem). *Assume the existence of one-way functions. Then there exists a constant-round resettably-sound zero-knowledge argument for all of NP.*

By relying on the above main theorem, we establish several other results on resettable security, by applying transformations from [11]: Assuming one-way functions, all of NP has

- a constant-round resettably-witness-indistinguishable argument of knowledge;
- a $\tilde{O}(\log n)$ -round resettable-zero-knowledge argument of knowledge.

(Roughly speaking, in a resettably-witness indistinguishable (resp., zero-knowledge) argument, the witness indistinguishability (resp., zero-knowledge) property is required to hold also in the presence of a resetting verifier.) For the above-mentioned primitives, previous results required additional cryptographic assumptions (the existence of collision-resistant hash-functions or oblivious transfer protocols). Subsequent works [33, 48], building upon our results, have shown how to construct the stronger notion of simultaneously resettable zero-knowledge argument for NP—simultaneous resettability here means that security (both zero-knowledge and soundness) holds even with respect to resetting attackers.

We emphasize that for all the above results, the use of non-black-box techniques are inherent. Our results lead to improvements also for cases when black-box simulation can be used: prior to our results, resettable zero-knowledge arguments (without the argument of knowledge property) were known only based on the existence of CRHs, but these protocols were actually proven secure using black-box simulation. As mentioned above, we are able to establish even the stronger notion of a resettable zero-knowledge argument of *knowledge* assuming only one-way functions.

2.1.2 Our Techniques

To explain our techniques, let us start by very briefly recalling the idea behind Barak’s constant-round public-coin protocol; we will then explain how this protocol is used to get a resettable-sound zero-knowledge protocol. The protocol relies on the existence of a family of collision-resistant hash function $h : \{0, 1\}^* \rightarrow \{0, 1\}^n$; note that any such family of collision-resistant hash functions can be implemented from a family of collision-resistant hash functions mapping n -bit string into $n/2$ -bit strings using *tree hashing* [95].

Roughly speaking, on common input 1^n and $x \in \{0, 1\}^{\text{poly}(n)}$, the Prover P and Verifier V , proceed in two stages. In Stage 1, V starts by selecting a function h from a family of collision-resistant hash function and sends it to P ; P next sends a commitment $c = \text{Com}(0^n)$ to a string of length n , and finally, V sends a “challenge” $r \in \{0, 1\}^{2^n}$. In Stage 2, P shows (using a witness indistinguishable argument of knowledge) that either x is true, or that c is a commitment to a “hash” (using h) of a program M (i.e., $c = \text{Com}(h(M))$) such that $M(c) = r$.

Roughly speaking, soundness follows from the fact that even if a malicious prover P^* tries to commit to (the hash of) some program M (instead of committing to 0^n), with high probability, the a string r sent by V will be different from $M(c)$ (since r is chosen independently of c). To prove ZK, consider the non-black-box simulator S that commits to a hash of the code of the malicious verifier V^* ; note that, by definition, it holds that $M(c) = r$, and the simulator can use the code of V^* as a “fake” witness in the final proof. To formalize this approach, the witness indistinguishable argument in Stage 2 must actually be a witness indistinguishable *universal argument* (WIUA) [96, 10] since the statement being proven is that c is a commitment to a program M of *arbitrary* polynomial-size such that $M(c) = r$ within some *arbitrary* polynomial time, and this statement is not in NP. Constant-round public-coin WIUAs are known based on the existence of CRH; as a result, the whole protocol is constant-round and public-coin, based on the existence of CRH.

Barak et al. [11] further show that any constant-round public-coin zero-knowledge argument of knowledge can be transformed into a resettably-sound zero-knowledge argument, by simply having the verifier generate its (random) message in every round by applying a pseudorandom function to the current partial transcript.³

Why hash functions are needed Note that hash functions are needed in two places in Barak’s protocol. First, since there is no *a-priori* polynomial upper-bound of the length of the code of V^* , we require the simulator to commit to the hash of the code of V^* . Secondly, since there is no *a-priori* polynomial upper-

³Strictly speaking, Barak’s protocol is not a argument of knowledge, but rather a “weak” argument of knowledge (see [10, 11] for more details), but the transformation of [11] applies also to such protocol.

bound on the running-time of V^* , we require the use of universal arguments (and such constructions are only known based on the existence of collision-resistant hash functions).

Using signature schemes instead of CRHs Our main idea is noticing that digital signature schemes—which can be constructed based on one-way functions—share many of the desirable properties of CRHs. In particular, we will show how to appropriately instantiate (a variant of) Barak’s protocol using signature schemes instead of using CRHs. Recall that “fixed-length” signature schemes, that allow signing messages of arbitrary polynomial-length (e.g length $2n$) using a length n signature, are known based on just one-way functions [113]. In fact, based on the same assumption, *strong* fixed-length signature schemes are known: in a strong signature scheme no polynomial time attacker can obtain a *new* signature even for messages that it has seen a signature on [66]. We observe that such signature scheme share a lot of properties with CRHs. First of all, they are compressing. More importantly, we observe that by the unforgeability requirement of strong signatures, no attacker can find a single valid signature σ for two distinct messages m, m' —that is, signatures satisfy a collision-resistance property. As will be seen later, this latter property crucially requires strong signatures, since plain signatures can have collisions without necessarily contradicting security. Additionally, by using an appropriate analog of tree hashing, a *signature tree* could be used to compress arbitrary length messages into a signature of length n .

Can we conclude here, and simply replace the CRHs in Barak’s protocol with strong, fixed-length, signature schemes? The problem with naively implementing this idea is that the collision-resistance property of strong signature schemes

only holds against an attacker that does *not* know the secret key. On the other hand, to generate signatures, knowledge of the secret key is needed. In our application, the simulator—acting as a prover—needs to be able to generate signature (in order to “hash down” the program, and in the universal argument) but at the same time, we need to ensure collision-resistance against cheating provers. So if we let the prover generate the signature keys, simulation is easy, but soundness no longer holds, whereas if we let the verifier generate the signature keys and only sends the verification key to the prover, then soundness holds, but it is no longer clear how to perform a simulation. We resolve this issue by using a “hybrid approach”: we have the verifier generate the signature keys, but also give the prover access to a *single* signing query. More precisely, in an initial stage of the protocol, the verifier generates a signature key-pair sk, vk and sends only the verification key vk to the prover. Next, in a “signature slot”, the prover sends a message m to the verifier, and the verifier computes and returns a valid signature σ of m (using sk). (We note that such a signature slot was previously used by [91] in a quite different context, but as we shall see shortly, some of their techniques will be useful also to us.) Finally, the protocol proceeds essentially as in Barak’s protocol, but where the CRH is replaced by the signature scheme. Implementing this is somewhat subtle: first, the statement proved in the WIUA in Barak’s protocol considers the hash function h (e.g., prover needs to prove statements of the type $h(m) = q$). In our approach since “hashing” has been replaced by “signing”, this would require the honest prover to prove things related to the secret-key (e.g., $\text{Sign}_{sk}(m) = q$), but the honest prover does not know sk . This issue is easily resolved by instead of letting the prover show that signatures used (as “hashes”) verify—i.e., that $\text{Ver}_{vk}(m) = q$. Another issue is that in Barak’s protocol, the honest prover actually needs to perform multiple

hashes to complete the WIUA, while we only allow the honest, non-rewinding prover to have access to a single signing (“hash”) query. We resolve this second issue by relying on an instantiation of Barak’s protocol due to Pass and Rosen [105], which relies on a special-purpose WIUA, in which the honest prover never needs to perform any hashing.⁴ Now completeness of this protocol follows in exactly the same way as in [7, 105].

For soundness, note that since the prover does not get to see sk , soundness follows in a similar way to Barak’s protocol. In fact, if the signature scheme used satisfies strong unforgeability, then the signature trees are collision-resistant with respect to attackers that get vk and *have access to a signing oracle*, and collision-resistance of the signature tree is the only property needed to prove soundness as in Barak’s protocol. (Note that we here only require collision-resistance with respect to attackers that get a single query to a signing oracle, but the more general result will be useful when we consider resettable-soundness.)

Let us turn to zero-knowledge. At first sight, it seems that we still have an issue. The prover just gets a single signature, but to complete the simulation, the simulator needs an a-priori unbounded polynomial number of signatures (to e.g., “hash down” a program of a-priori unbounded polynomial-size.⁵) Note, however, that the simulator can always *rewind* the verifier to get as many signatures as it wants and can thus complete the simulation in a similar way to the one used in Barak’s protocol. This approach doesn’t quite work: the malicious verifier V^* may not always agree to sign every message requested by the simulator; we deal with this issue in the same way as in [91], rather than having the simulator send the messages it wants to be signed in the clear, it simply sends

⁴In fact, an early version of Barak’s protocol also had this property.

⁵Also in the implementation of the WIUA, an a-priori unbounded number of “hashes” are needed.

a commitment to them. To make use of such a simulator strategy, we appropriately modify the notion of a signature tree to consist of signatures of commitments to signatures etc; we refer to this type of a signature tree as a “sig-com” tree.

So, we now have a zero-knowledge protocol that is based on one-way functions (and is constant-round). But it is no longer public-coin!

Nonetheless, let us still apply the transformation of [11] to the protocol (i.e., we have the verifier generate its random coins in each round by applying a PRF to the current partial transcript). We refer to this transformation as the PRF transformation. Clearly, the protocol is still zero-knowledge (since we only modified the verifier strategy). As it turns out, the resulting protocol is actually also resettably-sound: note that, except for the generation of the signature keys, the first verifier message containing the verification key, and the signature slot added in the beginning of the protocol, the protocol still is public-coin, and the same argument as in [68, 11] can be used to show that in the public-coin part of the protocol, rewindings do not “help” a resetting cheating prover. So, in essence, the only “advantages” a resetting prover gets is that it may rewind the signature slot, and thus get an arbitrary polynomial number of signatures on messages of its choice. But, as noted above, signature trees are collision-resistant even with respect to an attacker that gets an arbitrary polynomial number of queries to a signing oracle and thus resettable-soundness follows in exactly the same way as the (non-resetting) soundness property.

Beyond resettably-sound zero-knowledge For the applications of a) a constant-round resettably witness-indistinguishable argument of knowledge, and b) $\tilde{O}(\log n)$ -round resettable-zero-knowledge argument of knowledge for NP, we simply plug in our resettably-sound zero-knowledge argument of knowledge

into the protocols of [41, 11] with some minor modifications.

A PCP-free construction Just as the construction of Barak’s protocol, our constructions rely on universal arguments, which in turn rely on Probabilistically Checkable Proofs (PCPs). Intriguingly, the approach of Bitansky and Paneth [20] does not rely on PCPs; on the other hand, it relies on some other quite heavy machinery: “unobfuscatable functions” [13] and general secure two-party computation [69].

As we now sketch, our approach can be instantiated without the use of PCPs, and without introducing any other machinery. Recall that in Barak’s protocol the universal argument is used to prove a statement of the form c is a commitment to a hash of a program M such that $M(c) = r$. Also recall that (in the [105] variant of [7]) the honest prover never needs to engage in the universal argument, it is only the simulator that needs to prove the above statement. Rather than providing a universal argument, we let the simulator prove $M(c) = r$ in a *piecemeal* fashion, by making the verifier certify every step of the computation of M . This strategy is very similar to one employed in the “impossibility of instantiating random oracles” result of [42]⁶ (On a high-level, this type of piecemeal decomposition is also somewhat similar to what is done in the impossibility result of [13]; as such our approach brings out the connection between the techniques from [7] and [20].) More precisely, in the actual protocol, the verifier generates a key-pair vk', sk' for a signature scheme and sends vk' to the prover. The prover then provides the verifier with a commitment c_1 to a triple $(start_1, current_1, M_1)$, and a commitment c_2 to a triple $(start_2, current_2, M_2)$,

⁶The key difference is that the construction of [42] only considers an honest “non-aborting” verifier, whereas we need to deal with also malicious “aborting” verifiers. This issue is analogous to why we rely on “sig-com” trees (consisting of signatures of commitments to signatures etc.) as opposed to “plain” signature trees.

where $start_1, start_2, current_1$ and $current_2$ are compressed⁷ Turing Machine configurations, while M_1 and M_2 are Turing Machine descriptions. The prover also provides a witness indistinguishable argument of knowledge that either a) $x \in L$ or b) $start_2 = current_2$, corresponding to a starting configuration or c) $start_1 = start_2, M_1 = M_2$, and performing *one step* of computation given $current_1$ leads to $current_2$, and c_1 has been previously signed. (Note that since we use tree-hashing, verification of condition b) and c) can both be done in time polylogarithmic in the length of the configurations). If the argument of knowledge is accepting, the verifier signs c_2 . Roughly speaking, the above “slot” makes it possible for the simulator to get a signature on (commitments to signature-trees of) $(start, start, M)$, where $start$ is the initial configuration of $M(\sigma)$ (using condition b), and next by rewinding the verifier sufficiently many times to get signatures on later configurations $(start, current_t, M)$ in the computation of $M(\sigma)$ (using condition c). Thus, finally, the simulator can get a signature on $(start, final, M)$ where $final$ is the terminating configuration of the computation of $M(\sigma)$. The simulator can then use this signature to convince the verifier that $M(c) = r$ where M is the program committed to in c , as long as it committed to $M = V^*$.

A complete formalization appears in Section 2.6.

2.1.3 Subsequent Work

A very recent elegant work by Bitansky and Paneth [33] (developed subsequently to our results) shows an alternative approach for obtaining resettably-sound arguments (and related primitives) from one-way functions, by first con-

⁷We describe our protocol’s compression using collision resistant hash functions tree, but we may also instantiate tree-hashing with signature-trees to get an implementation based on one-way functions.

structuring functions that are “approximately” unobfuscatable, and relying on the connection between resettable-soundness and unobfuscatable functions from [20].

Also, subsequent works [33, 48] build upon our techniques and show how to get simultaneous resettable from only one-way functions.

2.1.4 Outline

In Section 2.3 we provide formal definitions of signature trees, and provide collision-resistance properties of such trees. To formalize our construction of resettable-sound zero-knowledge in a modular way, in Section 2.4, we first consider an “oracle-aided” model, in which players have access to a signing oracle. We first show that the universal argument construction of Barak and Goldreich [10] can be instantiated using one-way functions in such an oracle-aided model, by replacing “hashing” with “signing”. We next show how to instantiate Pass and Rosen’s [105] variant of Barak protocol in the same way (by relying on the oracle-aided construction of universal arguments). This leads to a constant-round oracle-aided public-coin zero-knowledge argument of knowledge, satisfying a key property: the honest prover never needs to access the oracle. We may next apply the transformation of [11] to this protocol to obtain an oracle-aided resettable-sound zero-knowledge argument of knowledge satisfying the same key property (the results of [11] relativize and thus we can directly apply them also to oracle-aided protocols).

In Section 2.5, we present a general transformation, transforming any oracle-aided resettable-sound zero-knowledge argument (of knowledge) satisfying

the above key property, into a resettably-sound zero-knowledge argument (of knowledge) in the “plain” model (i.e. without any oracle): the transformation simply consists of adding a signature slot at the beginning of the protocol. Taken together with our result in Section 2.4, this yields a constant-round resettably-sound zero-knowledge argument of knowledge for NP based on one-way functions.

In Section 2.6, we present an alternative construction of resettably sound zero knowledge, without using PCPs or UAs. As mentioned earlier, the construction relies on proving, piecemeal, a statement about a machine M underlying a prover’s commitment, using the help of a rewindable verified computation slot to validate individual steps of the computation of M .

In Section 2.7, we present applications such as resettable witness indistinguishable and resettable zero knowledge protocols.

2.2 Preliminaries

2.2.1 Notations

Let \mathcal{N} denote the set of positive integers, and $[n]$ denote the set $\{1, 2, \dots, n\}$. Given a string x , we let x_i denote the i th bit of x , and $x_{\leq i}$ denote the prefix of x upto and including its i th bit. By a probabilistic algorithm we mean a Turing machine that receives an auxiliary random tape as input. If M is a probabilistic algorithm, then for any input x , the notation “ $M_r(x)$ ” denotes the output of the M on input x when M ’s random tape is fixed to r , while $M(x)$ represents the dis-

tribution of outputs of $M_r(x)$ when r is chosen uniformly. An oracle algorithm is a machine that gets oracle access to another machine. Given a probabilistic oracle algorithm M and a probabilistic algorithm A , we let $M^A(x)$ denote the probability distribution over the outputs of the oracle algorithm M on input x , when given oracle access to A .

By $x \leftarrow \mathcal{S}$, we denote an element x is sampled from a distribution \mathcal{S} . If F is a finite set, then $x \leftarrow F$ means x is sampled uniformly from the set F . By $R \leftarrow \{0, 1\}^\infty$, we denote sampling an infinite random binary string R (for example, the random tape of a Turing Machine), by sampling each bit uniformly from $\{0, 1\}$. To denote the ordered sequence in which the experiments happen we use semicolon, e.g. $(x \leftarrow \mathcal{S}; (y, z) \leftarrow A(x))$. Using this notation we can describe probability of events. For example, if $p(\cdot, \cdot)$ denotes a predicate, then $Pr[x \leftarrow \mathcal{S}; (y, z) \leftarrow A(x) : p(y, z)]$ is the probability that the predicate $p(y, z)$ is true in the ordered sequence of experiments $(x \leftarrow \mathcal{S}; (y, z) \leftarrow A(x))$. The notation $\{(x \leftarrow \mathcal{S}; (y, z) \leftarrow A(x) : (y, z))\}$ denotes the resulting probability distribution $\{(y, z)\}$ generated by the ordered sequence of experiments $(x \leftarrow \mathcal{S}; (y, z) \leftarrow A(x))$.

2.2.2 Computational Indistinguishability

The following definition of computational indistinguishability originates in the seminal paper of Goldwasser and Micali [75]. Let X be a countable set of strings. A **probability ensemble indexed by X** is a sequence of random variables indexed by X . Namely, any element of $A = \{A_x\}_{x \in X}$ is a random variable indexed by X .

Definition 1 (Indistinguishability). *Let X and Y be countable sets. Two ensembles $\{A_{x,y}\}_{x \in X, y \in Y}$ and $\{B_{x,y}\}_{x \in X, y \in Y}$ are said to be **computationally indistinguishable over X** , if*

for every probabilistic machine D (the distinguisher) whose running time is polynomial in its first input, there exists a negligible function $\nu(\cdot)$ so that for every $x \in X, y \in Y$:

$$\left| \Pr [D(x, y, A_{x,y}) = 1] - \Pr [D(x, y, B_{x,y}) = 1] \right| < \nu(|x|)$$

(In the above expression, D is simply given a sample from $A_{x,y}$ and $B_{x,y}$, respectively.)

2.2.3 Interactive Arguments

Definition 2 (Interactive Arguments). A pair of interactive algorithms (P, V) is an **interactive argument** for a NP language L with witness relation R_L if it satisfies the following properties:

- *Completeness: There exists a negligible function $\mu(\cdot)$, such that for all $x \in L$, if $w \in R_L(x)$,*

$$\Pr[(P(w), V)(x) = 1] = 1 - \mu(|x|)$$

- *Soundness: For all non-uniform polynomial-time adversarial prover P^* , there exists a negligible function $\mu(\cdot)$ such that for every all $x \notin L$,*

$$\Pr[(P, V)(x) = 1] \leq \mu(|x|)$$

If the following condition holds, (P, V) is an **argument of knowledge**:

- *Argument of knowledge: There exists an expected PPT algorithm E such that for every polynomial-size P^* , there exists a negligible function $\mu(\cdot)$ such that for every x ,*

$$\Pr[w \leftarrow E(P^*, x); w \in R_L(x)] \geq \Pr[(P^*, V)(x) = 1] - \mu(|x|)$$

If E only makes use of its first argument P^* in a black-box way, i.e. only as a subroutine, then we call E a black-box extractor. Otherwise, we call E a non-black box extractor.

2.2.4 Witness Indistinguishability

An interactive protocol is **witness indistinguishable** (WI) [53] if the verifier’s view is “independent” of the witness used by the prover for proving the statement. In this context, we focus on languages $L \in \text{NP}$ with a corresponding witness relation \mathbf{R}_L . Namely, we consider interactions in which on common input x the prover is given a witness in $\mathbf{R}_L(x)$. For any adversarial verifier V^* , let $\text{View}_{V^*} \langle P(w), V(z) \rangle (x)$ be the random variable that denotes V^* ’s view in an interaction with P , when V^* is given auxiliary input z , P is given witness w , and both parties are given common input x .

Definition 3 (Witness-indistinguishability). *An interactive protocol (P, V) for $L \in \text{NP}$ is **witness indistinguishable** for \mathbf{R}_L if for every PPT adversarial verifier V^* , and for every two sequences $\{w_x^1\}_{x \in L}$ and $\{w_x^2\}_{x \in L}$, such that $w_x^1, w_x^2 \in \mathbf{R}_L(x)$ for every $x \in L$, the following ensembles are computationally indistinguishable over $x \in L$:*

$$\begin{aligned} & \{\text{View}_{V^*} \langle P(w_x^1), V^*(z) \rangle (x)\}_{x \in L, z \in \{0,1\}^*} \\ & \approx \{\text{View}_{V^*} \langle P(w_x^2), V^*(z) \rangle (x)\}_{x \in L, z \in \{0,1\}^*} \end{aligned}$$

A witness indistinguishable protocol may additionally be an argument of knowledge (WI-AOK), or a resettably-sound argument of knowledge (rsWI-AOK). Blum [23] gives a 3-round witness indistinguishable argument of knowledge for all of NP, while [11] show how to make it resettably-sound with the

same round-complexity. In the remainder of the paper, whenever we refer to WI-AOKs or rsWI-AOKs, we will mean the 3 round protocols provided by [23] and [11] respectively.

We additionally note that some WI-AOKs have the property that, given two proofs with the same first prover message but different verifier messages, one can combine these proofs to extract a witness for $x \in L$. The [23] WI-AOK has this property, which we will make use of.

2.2.5 Commitment Schemes

Commitment protocols allow a *sender* to commit itself to a value while keeping it secret from the *receiver*; this property is called **hiding**. At a later time, the commitment can only be opened to a single value as determined during the commitment protocol; this property is called **binding**. Commitment schemes come in two different flavors, statistically binding and statistically hiding; we only make use of statistically binding commitments in this paper. Below we sketch the properties of a statistically binding commitment; full definitions can be found in [66].

In statistically binding commitments, the binding property holds against unbounded adversaries, while the hiding property only holds against computationally bounded (non-uniform) adversaries. The statistical-binding property asserts that, with overwhelming probability over the randomness of the receiver, the transcript of the interaction fully determines the value committed to by the sender. The computational-hiding property guarantees that the commitments to any two different values are computationally indistinguishable.

Non-interactive statistically-binding commitment schemes can be constructed using any one-to-one one-way function (see Section 4.4.1 of [66]). Allowing some minimal interaction (in which the receiver first sends a single random initialization message), statistically-binding commitment schemes can be obtained from any one-way function [98, 80].

2.2.6 Zero Knowledge

We start by recalling the definition of zero knowledge from [76].

Definition 4 (Zero-knowledge [76]). *An interactive protocol (P, V) for language L is **zero-knowledge** if for every PPT adversarial verifier V^* (where running time is defined in terms of the length of the common input), there exists a PPT simulator S such that the following ensembles are computationally indistinguishable over $x \in L$:*

$$\{\text{View}_{V^*} \langle P(w), V^*(z) \rangle (x)\}_{x \in L, w \in R_L(x), z \in \{0,1\}^{\text{poly}(|x|)}} \approx \{S(x, z)\}_{x \in L, z \in \{0,1\}^{\text{poly}(|x|)}}$$

2.2.7 Resetably Sound Zero Knowledge

Let us recall the definition of resettable soundness due to [11].

Definition 5 (Resetably-sound Arguments [11]). *A resetting attack of a cheating prover P^* on a resettable verifier V is defined by the following two-step random process, indexed by a security parameter n .*

1. *Uniformly select and fix $t = \text{poly}(n)$ random-tapes, denoted r_1, \dots, r_t , for V , resulting in deterministic strategies $V^{(j)}(x) = V_{x,r_j}$ defined by $V_{x,r_j}(\alpha) = V(x, r_j, \alpha)$,*

where $x \in \{0, 1\}^n$, $j \in [t]$ and $V(x, r, \alpha)$ denotes the message sent by the strategy V on common input x , random-tape r , after seeing the message-sequence α . Each $V^{(j)}(x)$ is called an incarnation of V .

2. On input 1^n , machine P^* is allowed to initiate poly(n)-many interactions with the $V^{(j)}(x)$'s. The activity of P^* proceeds in rounds. In each round P^* chooses $x \in \{0, 1\}^n$ and $j \in [t]$, thus defining $V^{(j)}(x)$, and conducts a complete session with it.

Let (P, V) be an interactive argument for a language L . We say that (P, V) is a *resettably-sound argument for L* if the following condition holds:

- *Resettably-soundness*: For every polynomial-size resetting attack, the probability that in some session the corresponding $V^{(j)}(x)$ has accepted and $x \notin L$ is negligible.

We will also consider a slight weakening of the notion of resettably soundness, where the statement to be proven is fixed, and the verifier uses a single random tape (that is, the prover cannot start many independent instances of the verifier).

Definition 6 (Fixed-input Resettably-sound Arguments [109]). *An interactive argument (P, V) for a NP language L with witness relation R_L is **fixed-input resettably-sound** if it satisfies the following property: For all non-uniform polynomial-time adversarial prover P^* , there exists a negligible function $\mu(\cdot)$ such that for every $x \notin L$,*

$$\Pr[R \leftarrow \{0, 1\}^\infty; (P^{*V_R(x)}, V_R)(x) = 1] \leq \mu(|x|)$$

As the following claim shows, any zero-knowledge *argument of knowledge* satisfying the weaker notion can be transformed into one that satisfies the

stronger one, while preserving zero-knowledge (or any other secrecy property against malicious verifiers, for example, witness indistinguishability).

Claim 2. *Let (P, V) be a fixed-input resettably sound zero-knowledge (resp. witness indistinguishable) argument of knowledge for a language $L \in \text{NP}$. Then there exists a protocol (P', V') , with the same number of rounds, that is a (full-fledged) resettably-sound zero-knowledge (resp. witness indistinguishable) argument of knowledge for L .*

Proof. : We rely on the PRF transformation used in [11], but since we are dealing with private-coin protocols, we simply apply it to the random tape of the verifier. More precisely, the new verifier V' now chooses its random coins by applying a PRF to the statement x , and then continues its execution by simulating V using these random coins. (The honest prover remains unchanged). Since this change only modifies the verifier, the zero-knowledge property of (P, V) is preserved.

We prove the full-fledged resettable soundness of (P', V') by reducing to the single-instance resettable-soundness of (P, V) .

Let P^* be a malicious prover breaking the *full-fledged* resettable soundness of (P', V') with probability p_1 . Without loss of generality, we assume that P^* interacts with only a single verifier strategy $V' = V'^{(j)}$ for some j (e.g. the $V'^{(j)}$ for which it has the highest success in breaking resettable soundness), since all the other verifier strategies $V'^{(j')}$ for $j' \neq j$ use independent random tapes $r_{j'}$, and can thus be internally simulated by the malicious prover. This change reduces the success probability of P^* by at most a polynomial factor, to, say, p_2 .

We let $l = l(n)$ be a bound on the number of different x with which P^* incarnates V' . We consider a hybrid where instead of V' applying a PRF to x to gener-

ate its random tape, it uses a freshly sampled random tape r for each different x . This is equivalent to having l different incarnations V_1, \dots, V_l (the original verifier), each using independent random tapes. With probability $\geq p_3 = p_2 - \text{negl}(n)$, P^* convinces at least one V_i to accept an $x \notin L$. This is because, if $p_3 - p_2$ is non-negligible, then P^* breaks the resettable soundness of V' with a noticeably higher probability than that with which it breaks the resettable soundness of at least one of V_1, \dots, V_l , and further, this gap can be explicitly detected using the knowledge extractor for the protocol (an important point, as noted in [11]). Consequently, P^* could be used to distinguish between a PRF and a random function, contradicting the PRF's security.

Finally, there must be one index j such that P^* succeeds in convincing V_j with probability $\geq p_4 = p_3/l$, thus breaking single-instance resettable soundness of (P, V) . Since p_4 is at most a polynomial loss from p_1 , we conclude that single-instance resettable soundness of (P, V) implies full-fledged resettable soundness of (P', V') . \square

2.3 Signature Trees

In this section, we define an analogue of Merkle-hash trees using signature schemes. Towards this, we will rely on the existence of strong, fixed-length, deterministic secure signature schemes. Recall that in a strong signature scheme, no polynomial-time attacker having oracle access to a signing oracle can produce a valid message-signature pair, unless it has received this pair from the signing oracle. The signature scheme being fixed-length means that signatures of arbitrary (polynomial-length) messages are of some fixed polynomial length.

Deterministic signatures do not use any randomness in the signing process once the signing key has been chosen. In particular, once a signing key has been chosen, a message m will always be signed in the same way.

Definition 7 (Strong Signatures). *A strong, length- ℓ , signature scheme SIG is a triple $(\text{Gen}, \text{Sign}, \text{Ver})$ of PPT algorithms, such that*

1. *for all $n \in \mathcal{N}, m \in \{0, 1\}^*$,*

$$\Pr[(\text{sk}, \text{vk}) \leftarrow \text{Gen}(1^n), \sigma \leftarrow \text{Sign}_{\text{sk}}(m);$$

$$\text{Ver}_{\text{vk}}(m, \sigma) = 1 \wedge |\sigma| \leq \ell(n)] = 1$$

2. *for every non-uniform PPT adversary A , there exists a negligible function $\mu(\cdot)$ such that*

$$\Pr[(\text{sk}, \text{vk}) \leftarrow \text{Gen}(1^n), (m, \sigma) \leftarrow A^{\text{Sign}_{\text{sk}}(\cdot)}(1^n);$$

$$\text{Ver}_{\text{vk}}(m, \sigma) = 1 \wedge (m, \sigma) \notin L] \leq \mu(n),$$

where L denotes the list of query-answer pair of A 's query to its oracle.

Strong, length- ℓ , deterministic signature schemes with $\ell(n) = n$ are known based on the existence of OWFs; see [101, 113, 66] for further details. In the rest of this paper, whenever we refer to signature schemes, we always means strong, length- n deterministic signature schemes.

Let us first note that strong signatures satisfy a “collision-resistance” property.

Claim 3. *Let $\text{SIG} = (\text{Gen}, \text{Sign}, \text{Ver})$ be a strong (length- n) signature scheme. Then, for all non-uniform PPT adversaries A , there exists a negligible function $\mu(\cdot)$ such that*

for every $n \in \mathcal{N}$,

$$\Pr[(\text{sk}, \text{vk}) \leftarrow \text{Gen}(1^n), (m_1, m_2, \sigma) \leftarrow A^{\text{Sign}_{\text{sk}(\cdot)}}(1^n, \text{vk}); \\ \text{Ver}_{\text{vk}}(m_1, \sigma) = \text{Ver}_{\text{vk}}(m_2, \sigma) = 1] \leq \mu(n)$$

Proof. Assume for contradiction that there exists some non-uniform polynomial-time A such that A breaks “collision-resistance” property of SIG with probability $\frac{1}{p(n)}$ for infinitely many $n \in \mathcal{N}$, where p is a polynomial. We show that A can be used to break the strong unforgeability property of SIG. More precisely, note that if A outputs a valid signatures $(m_1, \sigma), (m_2, \sigma)$ without querying the signing oracle with m_1 and m_2 and receiving σ as a response to both queries, then A already breaks the strong security of the signature scheme (note that this property doesn’t hold if we do not use a strong signature scheme). Thus w.l.o.g. we may assume A queries both m_1 and m_2 to the signing oracle and receives σ as a response. We then simulate A , recording the previous messages queried to the oracle along with the responses. At each point during the execution of A , before forwarding the next query m to the oracle, we test if any of the previously received signatures are valid signatures for m . If so, we output m together with such a signature σ . Notice that if A always queries m_1 and m_2 and receives σ as a response, then we will intercept whichever of the two A queries second. Thus, for infinitely many n , with probability $\geq \frac{1}{p(n)}$, we forge a signature σ for some m before ever querying the signing oracle and receiving σ as a response. \square

We now define an analog of Merkle-hash tree which we call *signature trees* and show that they also satisfy a collision-resistant property. We index each node of a complete binary tree Γ of depth d by a binary string of length at most

d , where the root is indexed by the empty string λ , and each node indexed by γ has left and right children indexed $\gamma 0$ and $\gamma 1$, respectively.

Definition 8 (Signature Trees). *Let $\text{SIG} = (\text{Gen}, \text{Sign}, \text{Ver})$ be a strong, length- n signature scheme. Let (sk, vk) be a key-pair of SIG , and s be a string of length 2^d . A signature tree of the string s w.r.t. (sk, vk) is a complete binary tree of depth d , defined as follows.*

- A leaf l_γ indexed by $\gamma \in \{0, 1\}^d$ is set as the bit at position γ in s .
- An internal node l_γ indexed by $\gamma \in \bigcup_{i=0}^{d-1} \{0, 1\}^i$ satisfies that $\text{Ver}_{\text{vk}}((l_{\gamma 0}, l_{\gamma 1}), l_\gamma) = 1$.

Note that to *verify* whether Γ is a valid signature-tree of a string s w.r.t. the signature scheme SIG and the key-pair (sk, vk) knowledge of the secret key sk is not needed. However, to *create* a signature-tree for a string s , the secret key sk is needed.

The following notion of a signature path is the natural analog of an authentication path in a Merkle-tree.

Definition 9 (Signature Path). *A signature path w.r.t. SIG, vk and the root l_λ for the bit b at leaf $\gamma \in \{0, 1\}^d$ is a vector $\vec{\rho} = ((l_0, l_1), ((l_{\gamma_{\leq 1} 0}, l_{\gamma_{\leq 1} 1}), \dots, (l_{\gamma_{\leq d-1} 0}, l_{\gamma_{\leq d-1} 1})))$ such that for every $i \in \{0, \dots, d-1\}$, $\text{Ver}_{\text{vk}}((l_{\gamma_{\leq i} 0}, l_{\gamma_{\leq i} 1}), l_{\gamma_{\leq i}}) = 1$. Let $\text{PATH}^{\text{SIG}}(\vec{\rho}, b, \gamma, l_\lambda, \text{vk}) = 1 \iff \vec{\rho}$ is a signature path w.r.t. $\text{SIG}, \text{vk}, l_\lambda$ for b at γ .*

The following claim states that signature trees also satisfy an appropriate collision-resistance property: no non-uniform PPT attacker having oracle access to a signing oracle can output a root and valid signature paths for both 0 and 1 at some leaf γ .

Claim 4. Let $\text{SIG} = (\text{Gen}, \text{Sign}, \text{Ver})$ be a strong, length- n , signature scheme. Then, for every non-uniform PPT adversary A , there exists a negligible function μ such that:

$$\Pr[(\text{sk}, \text{vk}) \leftarrow \text{Gen}(1^n), (\vec{\rho}_0, \vec{\rho}_1, \gamma, l_\lambda) \leftarrow A^{\text{Sign}_{\text{sk}(\cdot)}}(1^n, \text{vk}); \\ \forall b \in \{0, 1\} \text{ PATH}^{\text{SIG}}(\vec{\rho}_b, b, \gamma, l_\lambda, \text{vk}) = 1] \leq \mu(n)$$

Proof. The claim directly follows from Claim 3 since any two valid signature-paths with the same root but different leaf value must contain a collision for the underlying signature scheme.

□

2.3.1 Sig-Com Schemes

For the technical reason explained in the introduction, we will rely on variant of signature trees consisting of alternating signatures and commitments. To formalize this, we consider the notion of a “sig-com” scheme:

Definition 10 (Sig-Com Schemes). Let $\text{SIG} = (\text{Gen}, \text{Sign}, \text{Ver})$ be a strong, length- n , signature scheme, and let Com be a non-interactive commitment schemes. Define $\text{SIG}' = (\text{Gen}', \text{Sign}', \text{Ver}')$ to be a triple of PPT machines defined as follows:

- $\text{Gen}' = \text{Gen}$.
- $\text{Sign}'_{\text{sk}}(m)$: compute a commitment $c = \text{Com}(m; \tau)$ using a uniformly selected τ , and let $\sigma = \text{Sign}_{\text{sk}}(c)$; output (σ, τ)
- $\text{Ver}'_{\text{vk}}(m, \sigma, \tau)$: Output 1 iff $\text{Ver}_{\text{vk}}(\text{Com}(m, \tau), \sigma) = 1$.

We call SIG' the Sig-Com Scheme corresponding to SIG and Com .

Note that the above definition of a sig-com scheme assumes that Com is a non-interactive commitment scheme. This is only for convenience of notation; the above definition, as well as all subsequent results directly apply also to 2-round commitment (i.e., families of non-interactive commitment schemes, as in [98]), by simply adding the first message q to the verification key of the sig-com scheme. As long as this first message is honestly generated, the binding security of the commitment still holds, and hiding holds even if it is not honestly generated. We note that in the rest of the paper, whenever we rely on the binding security of this commitment, the first message is always honestly generated.

Sig-com schemes also satisfy a collision-resistant property:

Claim 5 (Collision Resistance of Sig-Coms). *Let SIG = (Gen, Sign, Ver) be a strong, length- n signature scheme, Com be non-interactive commitment scheme, and let SIG' = (Gen', Sign', Ver') be a sig-com scheme corresponding to SIG and Com. Then, for any non-uniform PPT adversary A , there exists a negligible function μ such that for all $n \in \mathcal{N}$:*

$$\Pr[(\text{sk}, \text{vk}) \leftarrow \text{Gen}(1^n), (\sigma, m_1, m_2, \tau_1, \tau_2) \leftarrow A^{\text{Sign}_{\text{sk}(\cdot)}}(1^n, \text{vk}); \\ m_1 \neq m_2, \text{Ver}'_{\text{vk}}(m_1, \sigma, \tau_1) = \text{Ver}'_{\text{vk}}(m_2, \sigma, \tau_2) = 1] \leq \mu(n)$$

Proof. Note that by the binding property of Com, no non-uniform PPT can output a valid commitment c to two different messages $m_1 \neq m_2$ except with negligible probability. Thus, except with negligible probability, a successful non-uniform PPT attacker must output a signature for two different commitments $c_1 \neq c_2$, violating collision-resistance of SIG (i.e., Claim 3). \square

Note that in Claim 5, the attacker gets oracle access to a signature oracle (for SIG) as opposed to a sig-com oracle.

We may now define sig-com trees and sig-com paths in an analogous way to (plain) signature trees and paths.

Definition 11 (Sig-Com Trees). *Let $\text{SIG} = (\text{Gen}, \text{Sign}, \text{Ver})$ be a strong, length- n signature scheme, let Com be a non-interactive commitment and let $\text{SIG}' = (\text{Gen}', \text{Sign}', \text{Ver}')$ be the sig-com scheme corresponding to SIG and Com . Let (sk, vk) be a key-pair of SIG' , and s be a string of length 2^d . A signature tree of the string s w.r.t. (sk, vk) is a complete binary tree of depth d , defined as follows.*

- A leaf l_γ indexed by $\gamma \in \{0, 1\}^d$ is set as the bit at position γ in s .
- An internal node l_γ indexed by $\gamma \in \bigcup_{i=0}^{d-1} \{0, 1\}^i$ satisfies that there exists some τ_γ such that $\text{Ver}'_{\text{vk}}((l_{\gamma 0}, l_{\gamma 1}), l_\gamma, \tau_\gamma) = 1$.

Definition 12 (Sig-Com Path). *Let $\text{SIG}' = (\text{Gen}', \text{Sign}', \text{Ver}')$ be a sig-com scheme. A sig-com path w.r.t. SIG', vk and the root l_λ for the bit b at leaf $l_\gamma \in \{0, 1\}^d$ is a vector $\vec{\rho} = ((l_0, l_1, \tau_\lambda), ((l_{\gamma_{\leq 1} 0}, l_{\gamma_{\leq 1} 1}, \tau_{\gamma_{\leq 1}}), \dots, (l_{\gamma_{\leq d-1} 0}, l_{\gamma_{\leq d-1} 1}, \tau_{\gamma_{\leq d-1}}))$ such that for every $i \in \{0, \dots, d-1\}$, $\text{Ver}'_{\text{vk}}((l_{\gamma_{\leq i} 0}, l_{\gamma_{\leq i} 1}, l_{\gamma_{\leq i}}, \tau_{\gamma_{\leq i}})) = 1$. Let $\text{PATH}^{\text{SIG}'}(\vec{\rho}, b, \gamma, l_\lambda, \text{vk}) = 1$ if $\vec{\rho}$ is a signature path w.r.t. $\text{SIG}', \text{vk}, l_\lambda$ for b at l_γ .*

Sig-com trees also satisfy a collision-resistance property:

Claim 6. *Let $\text{SIG} = (\text{Gen}, \text{Sign}, \text{Ver})$ be a strong, length- n signature scheme, let Com be a non-interactive commitment and let $\text{SIG}' = (\text{Gen}', \text{Sign}', \text{Ver}')$ be the sig-com scheme corresponding to SIG and Com . Then, for every non-uniform PPT adversary A , there exists a negligible function μ such that:*

$$\Pr[(\text{sk}, \text{vk}) \leftarrow \text{Gen}(1^n), (\vec{\rho}_0, \vec{\rho}_1, \gamma, l_\lambda) \leftarrow A^{\text{Sign}_{\text{sk}(\cdot)}}(1^n, \text{vk});$$

$$\forall b \in \{0, 1\} \text{ PATH}^{\text{SIG}'}(\vec{\rho}_b, b, \gamma, l_\lambda, \text{vk}) = 1 \leq \mu(n)$$

Proof. As in Claim 4, the claim follows directly from Claim 5 since any two valid sig-com paths with the same root but different leaf values must contain a collision for the underlying sig-com scheme. \square

Canonical Sig-com Schemes Throughout the rest of the paper, we consider sig-com schemes SIG' and sig-com trees corresponding to a strong, length- n deterministic signature scheme SIG and a non-interactive commitment Com that generates n^2 bits long commitments to $2n$ bits strings. Thus, each node of the sig-com tree is an n -bit signature of an n^2 bits commitment of the two signatures of the children nodes. Hereafter, we refer to such a SIG' as a **canonical sig-com scheme**.

2.4 Oracle-Aided Resettably-sound Zero Knowledge Protocols

In this section we show how to construct a resettably-sound ZK argument in an oracle-aided model where the prover and verifier additionally have access to a public parameter generated prior to the interaction (in our protocol, this will be the verification key for a signature scheme), and, further the prover has access to an oracle, also generated prior to the interaction (in our protocol, this will be a signature/sig-com oracle).

More formally, let O be a probabilistic algorithm that on input a security parameter n , outputs a polynomial-length (in n) public-parameter pp , as well as the description of an oracle O . The oracle-aided execution of an interactive protocol with common input x between a prover P with auxiliary input y and a verifier V consist of first generating $pp, O \leftarrow O(1^{n^2})$ and then letting $P^O(x, y, pp)$

interact with $V(x, \text{pp})$.

Definition 13 (Oracle-aided Interactive Arguments). *A pair of oracle algorithms (P, V) is an O -oracle aided argument for a NP language L with witness relation R_L if it satisfies the following properties:*

- *Completeness: There exists a negligible function $\mu(\cdot)$, such that for all $x \in L$, if $w \in R_L(x)$,*

$$\Pr[\text{pp}, O \leftarrow O(1^{|x|}); (P^O(w), V)(x, \text{pp}) = 1] \geq 1 - \mu(|x|)$$

- *Soundness: For all non-uniform polynomial-time adversarial prover P^* , there exists a negligible function $\mu(\cdot)$ such that for every all $x \notin L$,*

$$\Pr[\text{pp}, O \leftarrow O(1^{|x|}); (P^{*O}, V)(x, \text{pp}) = 1] \leq \mu(|x|)$$

Additionally, if the following condition holds, (P, V) is an O -oracle aided argument of knowledge:

- *Argument of knowledge: There exists a expected PPT algorithm E such that for every polynomial-size P^* , there exists a negligible function $\mu(\cdot)$ such that for every x ,*

$$\begin{aligned} & \Pr[\text{pp}, O \leftarrow O(1^{|x|}); w \leftarrow E^O(P^*, x, \text{pp}); w \in R_L(x)] \\ & \geq \Pr[\text{pp}, O \leftarrow O(1^{|x|}); (P^{*O}, V)(x, \text{pp}) = 1] - \mu(|x|) \end{aligned}$$

If E uses its first argument P^ only in a black-box way, i.e. only as a subroutine, then E is called a black-box extractor. Otherwise, E is called a non-black box extractor.*

Definition 14 (Oracle-aided Resettable-sound Interactive Arguments). *An \mathcal{O} -oracle aided resetting attack of a cheating prover P^* on a resettable verifier V is defined by the following three-step random process, indexed by a security parameter n .*

1. *An initial setup where a public parameter and an oracle are generated: $\text{pp}, \mathcal{O} \leftarrow \mathcal{O}(1^n)$. P^* is given pp and oracle access to \mathcal{O} .*
2. *Uniformly select and fix $t = \text{poly}(n)$ random-tapes, denoted r_1, \dots, r_t , for V , resulting in deterministic strategies $V^{(j)}(x) = V_{\text{pp},x,r_j}$ defined by $V_{\text{pp},x,r_j}(\alpha) = V(\text{pp}, x, r_j, \alpha)$, where $x \in \{0, 1\}^n$ and $j \in [t]$. Each $V^{(j)}(x)$ is called an incarnation of V .*
3. *On input 1^n , machine P^* is allowed to initiate $\text{poly}(n)$ -many interactions with the $V^{(j)}(x)$'s. The activity of P^* proceeds in rounds. In each round P^* chooses $x \in \{0, 1\}^n$ and $j \in [t]$, thus defining $V^{(j)}(x)$, and conducts a complete session with it.*

Let (P, V) be an \mathcal{O} -oracle aided interactive argument for a language L . We say that (P, V) is an \mathcal{O} -oracle aided resettable-sound argument for L if the following condition holds:

- *\mathcal{O} -oracle aided resettable soundness: For every polynomial-size resetting attack, the probability that in some session the corresponding $V^{(j)}(x)$ has accepted and $x \notin L$ is negligible.*

Towards our goal of constructing of oracle-aided resettable-sound zero-knowledge, we now define and construct an oracle-aided version of universal arguments.

2.4.1 Oracle-aided Universal Arguments

Universal arguments (introduced in [10] and closely related to CS-proofs [87, 96]) are used in order to provide “efficient” proofs to statements of the form $y = (M, x, t)$, where y is considered to be a true statement if M is a non-deterministic machine that accepts x within t steps. The corresponding language and witness relation are denoted $L_{\mathcal{U}}$ and $\mathbf{R}_{\mathcal{U}}$ respectively, where the pair $((M, x, t), w)$ is in $\mathbf{R}_{\mathcal{U}}$ if M (viewed here as a two-input deterministic machine) accepts the pair (x, w) within t steps. Notice that every language in NP is linear time reducible to $L_{\mathcal{U}}$. Thus, a proof system for $L_{\mathcal{U}}$ allows us to handle all NP-statements. In fact, a proof system for $L_{\mathcal{U}}$ enables us to handle languages that are presumably “beyond” NP, as the language $L_{\mathcal{U}}$ is NEXP-complete (hence the name universal arguments).⁸ We here provide an oracle-aided variant of the [10] definition of universal arguments.

Definition 15 (Oracle-aided Universal argument). *An oracle-aided interactive argument (P, V) is called an \mathcal{O} -oracle-aided universal argument system if it satisfies the following properties:*

- **Efficient verification:** *There exists a polynomial p such that for any $y = (M, x, t)$, and for any pp, O generated by \mathcal{O} , the total time spent by the (probabilistic) verifier strategy V , on common input y, pp , is at most $p(|y| + |\text{pp}|)$. In particular, all messages exchanged in the protocol have length smaller than $p(|y| + |\text{pp}|)$.*
- **Completeness by a relatively efficient oracle-aided prover:** *For every $(y = (M, x, t), w)$ in $\mathbf{R}_{\mathcal{U}}$,*

$$\Pr[\text{pp}, O \leftarrow \mathcal{O}(1^{|y|}); (P^O(w), V)(y, \text{pp}) = 1] = 1.$$

⁸Furthermore, every language in NEXP is polynomial-time (but not linear-time) reducible to $L_{\mathcal{U}}$

Furthermore, there exists a polynomial q such that the total time spent by $P^O(w)$, on common input $y = (M, x, t)$, pp , is at most $q(T_M(x, w) + |\text{pp}|) \leq q(t + |\text{pp}|)$, where $T_M(x, w)$ denotes the running time of M on input (x, w) .

- Weak proof of knowledge for adaptively chosen statements: For every polynomial p there exists a polynomial p' and a probabilistic polynomial-time oracle machine E such that the following holds: for every non-uniform polynomial-time oracle algorithm P^* , if

$$\Pr[\text{pp}, O \leftarrow O(1^n); R \leftarrow \{0, 1\}^\infty; y \leftarrow P_R^{*O}(\text{pp}) : \\ (P_R^{*O}(y, \text{pp}), V(y, \text{pp})) = 1] > 1/p(n)$$

then

$$\Pr[\text{pp}, O \leftarrow O(1^n); R, r \leftarrow \{0, 1\}^\infty; y \leftarrow P_R^{*O}(\text{pp}) : \\ \exists w = w_1, \dots, w_t \in \mathbf{R}_U(y) \text{ s.t. } \forall i \in [t], \\ E_r^{P^*O}(\text{pp}, y, i) = w_i] > \frac{1}{p'(n)}$$

where $\mathbf{R}_U(y) \stackrel{\text{def}}{=} \{w : (y, w) \in \mathbf{R}_U\}$.

We note that we allow oracles O generated by O to be randomized, and require that the properties listed above hold even in the case of these randomized oracles.

Note that our proof of knowledge condition is somewhat different from the one used in [10] in that we allow the (cheating) prover to *adaptively* choose the statement to be proved, after having seen the public parameter, and having interacted with its oracle.

Nevertheless, as we shall see, the construction of [10] and their analysis will be useful to us. Recall that in the construction of [10] tree hashing is used to

hash down a “long” PCP proof into a fixed-length “tree root”; the soundness property relies on collision resistance of this tree hashing. Let SIG' be a canonical sig-com scheme with $\text{SIG} = (\text{Gen}, \text{Sign}, \text{Ver})$ and Com being its underlying signature scheme and commitment scheme. We observe that if we replace the use of tree hashing in [10] scheme with a sig-com tree using SIG' , then the resulting protocol is an \mathcal{O}^{SIG} -aided universal argument for the following signature oracle \mathcal{O}^{SIG} .

Definition 16 (Signature Oracle). *A signature oracle \mathcal{O}^{SIG} is defined as follows: On input a security parameter n , $\mathcal{O}^{\text{SIG}}(1^n)$ generates $(\text{vk}, \text{sk}) \leftarrow \text{Gen}(1^n)$ and lets $\text{pp} = \text{vk}$ and $\mathcal{O}(m) = \text{Sign}_{\text{sk}}(m)$ for every $m \in \{0, 1\}^{\text{poly}(n)}$.*

In fact, the universal argument has an even stronger completeness property that will be useful for us: completeness hold even if the prover only gets access to a sig-com oracle (instead of a signature oracle), and even if this is an *arbitrary* (not necessarily using the honest sign and commit algorithms) sig-com oracle, as long as the oracle outputs valid sig-com’s (for messages of a certain fixed length) with overwhelming probability. More formally,

Definition 17 (Valid Sig-com Oracle). *A randomized oracle \mathcal{O}' is a **valid** (SIG', ℓ) oracle if there is a negligible $\mu(\cdot)$ such that for every $n \in \mathbb{N}$, the following holds with probability $1 - \mu(n)$ over pp , $\mathcal{O} \leftarrow \mathcal{O}'(1^n)$: for every $m \in \{0, 1\}^{\ell(n)}$, $\mathcal{O}(m)$ returns (σ, τ) such that $\text{Ver}'_{\text{vk}}(m, \sigma, \tau) = 1$ with probability at least $1 - \mu(n)$.*

We note that oracles that use arbitrarily biased randomness for commitment are also considered *valid* sig-com oracles. (These are precisely the kind of oracles we will be forced to use later on).

Definition 18. *An \mathcal{O}^{SIG} -aided universal argument (P, V) has (SIG', ℓ) -completeness if*

there exists a prover P' such that the completeness condition holds for (P', V) when the oracle O^{SIG} is replaced by any valid (SIG', ℓ) oracle O' .

We now have the following theorem.

Theorem 7. *Let SIG' be a canonical sig-com scheme with SIG and Com being its underlying signature scheme and commitment scheme. Then there exists a polynomial ℓ and a (SIG', ℓ) -complete O^{SIG} -aided universal argument Π .*

The proof of the theorem identically follows that of Barak and Goldreich [10], with a minor modification to deal with adaptively chosen statements when proving the weak argument of knowledge property.

Proof. We construct such a universal argument in Fig. 2.1, which is essentially identical to the construction of [10], except that the Merkle hash tree is replaced by a sig-com tree. Note that both the efficient verification property and the completeness property (with a relatively efficient prover) follow by inspection. Furthermore, note that the (SIG', ℓ) -completeness holds as well, since the prover P only need to access an arbitrary valid (SIG', ℓ) oracle to produce the sig-com tree. (Here $l(n)$ is simply the length of the messages to which sig-coms are applied, i.e. $2n$ when SIG is a canonical sig-com scheme).

It remains to prove the weak proof of knowledge property (for adaptively chosen statements). Our proof is almost identical to that given by Barak and Goldreich in Section 3 of [10]. In fact, if we simply replace hash trees with sig-com trees and following their argument exactly, we have the following lemma:

Lemma 8 (implicit in Lemma 3.5 of [10]). *Let (P, V) be the O^{SIG} -aided protocol defined in Fig. 2.1. For every polynomial p , there exist oracle PPT algorithms E and CF*

Common Input: An instance $y = (M, x, t)$ of $L_{\mathcal{U}}$; let $n := |y|$.

Auxiliary input to prover: w such that $(y, w) \in \mathbf{R}_{\mathcal{U}}$ holds.

Primitives used:

- A PCP scheme for $L_{\mathcal{U}}$ with auxiliary properties as defined in [10], where
 - $P_{\text{PCP}}(y, w)$ generates a PCP proof π for $(y, w) \in \mathbf{R}_{\mathcal{U}}$.
 - V_{PCP} is the non-adaptive verifier for the PCP system, which makes m queries to the PCP proof.
 - $Q_{\text{PCP}}(y, r, d, i)$ generates the i th query of V_{PCP} with random tape r and common input y , when the hashed proof has depth d .
- A canonical sig-com scheme SIG' with SIG and Com as the underlying signature and commitment schemes; let \mathcal{O}^{SIG} be the corresponding signature oracle.

Set Up: Run $(\text{pp}, \mathcal{O}) \leftarrow \mathcal{O}^{\text{SIG}}(1^n)$, add pp to common input for P and V . Further, allow P oracle access to \mathcal{O} .

Protocol:

P_1 : Generate $\pi \leftarrow P_{\text{PCP}}(y, (w, 1^{t'}))$, where t' is the runtime of M on input (x, w) . Use \mathcal{O} to generate a sig-com tree for π w.r.t. pp , recording sig-com paths for each leaf. Send (d, l_λ) , the depth and the root of the sig-com tree, to V .

V_1 : Uniformly select randomness r for V_{PCP} , and send it to P .

P_2 : Generate queries $\{q_i\}_{i \in [m]}$ by using $Q_{\text{PCP}}(y, r, d, i)$ to generate the i th query for every $i \in [m]$. Generate sig-com paths $\{\vec{\rho}_i\}_{i \in [m]}$ for the bits $\{b_i = \pi_{q_i}\}_{i \in [m]}$ of π in the sig-com tree. Send the bits $\{b_i\}_{i \in [m]}$ together with the sig-com paths $\{\vec{\rho}_i\}_{i \in [m]}$ to V .

V accepts when:

- $\text{PATH}^{\text{SIG}'}(\vec{\rho}_i, b_i, q_i, l_\lambda, \text{pp}) = 1$ for every $i \in [m]$.
- V_{PCP} accepts when receiving $\{b_i\}_{i \in [m]}$ as the responses to its oracle queries.

Figure 2.1: An \mathcal{O}^{SIG} -aided Universal Argument.

and a polynomial q such that for every $n \in \mathcal{N}$ and every non-uniform PPT adversary P^* , if

$$\Pr[\text{pp}, O \leftarrow O(1^n); R \leftarrow \{0, 1\}^\infty; y \leftarrow P_R^{*O}(\text{pp}) : \\ (P_R^{*O}(\text{pp}), V(y, \text{pp})) = 1] > 1/p(n),$$

then with probability at least $1/q(n)$ over $(\text{pp}, O, R) \leftarrow O(1^n) \times \{0, 1\}^\infty$, it holds that either

$$\Pr[r \leftarrow \{0, 1\}^\infty; y \leftarrow P_R^{*O}(\text{pp}) : \exists w = w_1, \dots, w_t \in \mathbf{R}_u(y) \\ \text{s.t. } \forall i \in [t], E_r^{P_R^{*O}(\text{pp})}(\text{pp}, y, i) = w_i] > \frac{1}{q(n)},$$

or

$$\Pr[(\vec{\rho}_0, \vec{\rho}_1, \gamma, l_\lambda) \leftarrow \text{CF}^{P_R^{*O}(\text{pp})}(\text{pp}); \forall b \in \{0, 1\} \\ \text{PATH}^{\text{SIG}'}(\vec{\rho}_b, b, \gamma, l_\lambda, \text{pp}) = 1] \geq 1/q(n)$$

Given the above lemma, we observe that, for any P^{*O} , except for finitely many $n \in \mathcal{N}$, the latter condition can only hold with probability at most $1/2q(n)$ over $(\text{pp}, O, R) \leftarrow O(1^n) \times \{0, 1\}^\infty$. Otherwise, we will be able to use $\text{CF}^{P_R^{*O}}$ as an oracle aided adversary that succeeds in breaking the sig-com tree collision resistance of SIG' for infinitely many $n \in \mathcal{N}$ with probability $\geq 1/2(q(n))^2$ over $O(1^n), R$, and the randomness of CF .

Thus, assuming that SIG' is a secure sig-com scheme, the former condition of the lemma must hold with probability $\geq 1/2q(n)$ over $(\text{pp}, O, R) \leftarrow O(1^n) \times \{0, 1\}^\infty$ for all but finitely many $n \in \mathcal{N}$.

$$\Pr[\text{pp}, O \leftarrow O(1^n); R, r \leftarrow \{0, 1\}^\infty; y \leftarrow P_R^{*O}(\text{pp}) : \exists w = w_1, \dots, w_t \in \mathbf{R}_u(y) \text{ s.t.} \\ \forall i \in [t], E_r^{P_R^{*O}(\text{pp})}(\text{pp}, y, i) = w_i] > \frac{1}{2(q(n))^2}$$

Setting $p'(n) = 2(q(n))^2$, we have that E is an extractor for the weak proof of knowledge property. \square

2.4.2 Oracle-aided Zero-Knowledge Protocols

We now turn to constructing oracle-aided resettably-sound zero-knowledge protocols. We start by defining a strong notion of an \mathcal{O} -oracle-aided version of ZK. First of all, we restrict to protocols where the honest prover does not access the oracle. Secondly, we require that simulation can be performed given oracle access to *any* valid SIG' oracle. These two restrictions will be important when we later instantiate the oracle-aided protocol in the plain model.

Definition 19 (Oracle-aided Zero-Knowledge). *An interactive argument (P, V) is (SIG', ℓ) -oracle aided zero-knowledge for a NP language L with witness relation R_L if for every polynomial-time adversarial verifier V^* , there exists a simulator S , such that for every valid (SIG', ℓ) -oracle \mathcal{O}' , the following ensembles are indistinguishable over $x \in L$,*

$$\begin{aligned} & \{\text{pp}, \mathcal{O} \leftarrow \mathcal{O}'(1^{|x|}) : (\text{pp}, \text{View}_{V^*}(P(w), V^*(z))(x, \text{pp}))\}_{x \in L, w \in R_L(x), z \in \{0,1\}^*} \\ & \approx \{\text{pp}, \mathcal{O} \leftarrow \mathcal{O}'(1^{|x|}) : (\text{pp}, S^{\mathcal{O}}(x, z, \text{pp}))\}_{x \in L, w \in R_L(x), z \in \{0,1\}^*} \end{aligned}$$

(Note that in the above definition, the oracle is only used by the *simulator*, in order to prove the zero-knowledge property. The prover, however, may not get access to this oracle during the honest execution of the protocol. In fact, for our construction we require completeness of the protocol to hold without the prover having access to an oracle.)

We now turn to the question of constructing a protocol that is resettably

sound, and is also oracle-aided zero-knowledge. Note that, as a first attempt, we could try constructing a constant-round public-coin ZK protocol by replacing the tree hashing in Barak’s protocol [7] with sig-com trees, and then apply the PRF transformation of [11] to achieve resettable soundness. While this indeed could be used to get a resettably-sound ZK protocol in the oracle-aided model, the resulting protocol would require the honest prover to make polynomially many queries to the oracle (to complete the WIUA), and we want the prover to complete. To get around this, we instead rely on a variant of Barak’s protocol used in Pass and Rosen [105], which provides a “special-purpose” implementation of the WIUA used in Barak’s protocol in which the honest prover does not need to perform any “hashing”.⁹

More precisely, our protocol proceeds as follows. In Stage 1, P sends a commitment $c = \text{Com}(0^{2n})$, and then V sends back a challenge $r \in \{0, 1\}^n$ as in Barak’s protocol. In Stage 2, P and V first execute an “encrypted” universal argument $(P_{\text{UA}}, V_{\text{UA}})$ of the statement that “ c is a commitment to a sig-com tree root of a program M and $M(c) = r$,” where instead of sending the message in the clear, the prover sends commitments to the messages. The honest prover simply sends commitments to 0 (and thus will fail in this encrypted universal argument). Finally, P and V execute a witness-indistinguishable argument of knowledge of the statement that “ $x \in L$ OR V_{UA} accepts in the encrypted universal argument.

A formal description of the protocol can be found in Fig. 2.2 and Fig. 2.3. Note that, in this construction, the honest prover P can convince the verifier by proving $x \in L$ in the final witness indistinguishable argument without making any oracle queries. This leads to the following theorem. The proof of the

⁹In fact, early versions of Barak’s protocol also relied on such a special-purpose implementation of WIUA.

Common Input: An instance x of a language $L \in \text{NP}$ with witness relation \mathbf{R}_L .

Auxiliary input to P : A witness w such that $(x, w) \in \mathbf{R}_L$.

Primitives Used: A canonical sig-com scheme SIG' with SIG and Com as the underlying signature and commitment schemes, and a (SIG', ℓ) -complete \mathcal{O}^{SIG} -aided universal argument $(P_{\text{UA}}, V_{\text{UA}})$ with $\ell(n) = 2n$.

Set Up: Run $(\text{pp}, O) \leftarrow \mathcal{O}^{\text{SIG}}(1^n)$, add pp to common input for P and V . Furthermore, allow P oracle access to O .

Stage One (Trapdoor):

P_1 : Send $c_0 = \text{Com}(0^{2n}, \tau_0)$ to V with uniform τ_0

V_1 : Send $r \xrightarrow{\$} \{0, 1\}^n$ to P

Stage Two (Encrypted Universal Argument):

P_2 : Send $c_1 = \text{Com}(0^{2n}, \tau_1)$ for uniformly selected τ_1

V_3 : Send r' , uniformly chosen random tape for V_{UA}

P_3 : Send $c_2 = \text{Com}(0^k, \tau_2)$ for uniformly selected τ_2 , where k is the length of P_{UA} 's second message.

Stage Three: (Main Proof)

$P \Leftrightarrow V$: A WI-AOK $\langle P_{\text{WI}}, V_{\text{WI}} \rangle$ proving the OR of the following statements:

1. $\exists w \in \{0, 1\}^{\text{poly}(|x|)}$ s.t. $(x, w) \in \mathbf{R}_L$.
2. $\exists \langle p_1, p_2, \tau_1, \tau_2 \rangle$ s.t. $(\langle c_0, r, c_1, c_2, r', \text{pp} \rangle, \langle p_1, p_2, \tau_1, \tau_2 \rangle) \in \mathbf{R}_{L_2}$ (defined in Fig. 2.3).

Figure 2.2: \mathcal{O}^{SIG} -aided ZK Argument of Knowledge.

theorem closely follows [7, 105] but the proof of the “argument of knowledge” property requires special care to deal with the fact that a cheating prover may adaptively choose the statements to be proved in the encrypted universal argument (after having interacted with its oracle).¹⁰

Theorem 9. *Let SIG' be a canonical sig-com scheme with SIG and Com being its underlying signature scheme and commitment scheme. Then there exists an \mathcal{O}^{SIG} -oracle*

¹⁰In [7, 105] this issue does not arise since different, independently chosen hash-functions are used in Stage 1 and in Stage 2.

Relation 1: Let SIG' a sig-com scheme, with underlying signature scheme SIG and commitment scheme Com . Let ECC be a binary error-correcting code with constant min-distance and efficient encoding algorithm. We say that $\langle c_0, r, \text{pp} \rangle \in L_1$ if $\exists \langle \tau_0, d, l_\lambda, C, \{\vec{\rho}_i\}_{i \in [2^d]} \rangle$ such that

- $c_0 = \text{Com}((d, l_\lambda), \tau_0)$
- (d, l_λ) are the depth and root of a sig-com tree for C w.r.t. pp
- Each $\vec{\rho}_i$ is a valid sig-com path for leaf i of this sig-com tree. That is, $\text{PATH}^{\text{SIG}'}(\vec{\rho}_i, C_i, i, l_\lambda, \text{pp}) = 1$ for each i .
- $C = \text{ECC}(\Pi)$ for some circuit Π
- $\Pi(c_0) = r$.

We let \mathbf{R}_{L_1} be the witness relation corresponding to L_1 .

Relation 2: Let L_1 be described as above, with respect to SIG' and ECC . Let $(P_{\text{UA}}, V_{\text{UA}})$ be a (SIG', ℓ) -complete \mathcal{O}^{SIG} -aided universal argument with $\ell(n) = 2n$. We say that $\langle c_0, r, c_1, c_2, r', \text{pp} \rangle \in L_2$ if $\exists \langle p_1, p_2, \tau_1, \tau_2 \rangle$ such that

- $c_1 = \text{Com}(p_1, \tau_1), c_2 = \text{Com}(p_2, \tau_2)$.
- (p_1, r', p_2) constitutes an accepting $(P_{\text{UA}}, V_{\text{UA}})$ transcript for $\langle c_0, r \rangle \in L_1$.

We let \mathbf{R}_{L_2} be the witness relation corresponding to L_2 .

Figure 2.3: Relations used in the \mathcal{O}^{SIG} -aided ZK protocol in Fig. 2.2.

aided argument of knowledge (P, V) for NP; additionally,

1. (P, V) is constant-round and public-coin;
2. P does not make any queries to its oracle;
3. (P, V) is (SIG', ℓ) -oracle-aided zero-knowledge for $\ell(n) = 2n$.

Proof. We show that the construction provided in Fig. 2.2 and 2.3 satisfies the desired properties. By inspection, (P, V) is constant-round and public-coin, and P does not make any queries to its oracle. For the (SIG', ℓ) -oracle-aided zero-knowledge property, we construct a simulator identically to [105]. In brief, the straight-line simulator S will provide a proof to V^* using the second “trapdoor”

witness for Stage Three. It will do so using the oracle O to produce a sig-com tree for $\text{ECC}(\Pi)$ in Stage One with $\Pi = V^*$, and also to complete the encrypted UA in Stage Two. We further observe that the ZK simulator will work even with any valid (SIG', ℓ) -oracle, since such an oracle is sufficient to produce a correct sig-com tree in Stage One, and to complete the (SIG', ℓ) -oracle complete UA in Stage Two.

It remains to show the argument of knowledge property. We start by constructing a knowledge extractor E for (P, V) . $E(x, \text{pp})$ proceeds as follows: Given oracle access to a malicious prover $P^{*O}(x, \text{pp})$, E internally emulates the role of the honest verifier V for $P^{*O}(x, \text{pp})$ up to the beginning of Stage Three (i.e., the beginning of WI-AOK). Let α denote the partial transcript, and $P^{*O}(x, \text{pp}; \alpha)$ be the “residual” prover. Then E applies the witness extractor E_{WI} for (P_{WI}, V_{WI}) on $P^{*O}(x, \text{pp}; \alpha)$, and outputs E_{WI} 's output. Note that since $O \leftarrow O^{\text{SIG}}$ is efficient, $P^{*O}(x, \text{pp}; \alpha)$ is a polynomial size adversary in the plain model.

Clearly by inspection, E runs in expected polynomial time. Let ε be the success probability of P^* in convincing V . We first show that E_{WI} outputs a *valid* witness (either a *true* witness $w \in \mathbf{R}_L(x)$ or a *false* witness $(p_1, p_2, \tau_1, \tau_2) \in \mathbf{R}_{L_2}(c_0, r, c_1, c_2, r', \text{pp})$) with probability $\varepsilon - \text{negl}(|x|)$.

Let $\varepsilon(\text{pp}, O, \alpha) = \Pr[(P^{*O}, V_{WI})(x, \text{pp}; \alpha) = 1]$, i.e., the probability that the residual prover $P^{*O}(x, \text{pp}; \alpha)$ convinces V_{WI} in Stage Three. By definition, $\mathbb{E}_{\text{pp}, O, \alpha}[\varepsilon(\text{pp}, O, \alpha)] = \varepsilon$. By the argument of knowledge property of the WI-AOK (P_{WI}, V_{WI}) , there exists a negligible μ such that for all x, pp, α $E_{WI}^{P^{*O}(x, \text{pp}; \alpha)}$ outputs a valid witness with probability at least $\varepsilon(\text{pp}, O, \alpha) - \mu(|x|)$. It follows that in the execution of E , E_{WI} outputs a valid witness with probability at least $\mathbb{E}_{\text{pp}, O, \alpha}[\varepsilon(\text{pp}, O, \alpha) - \mu(|x|)] \geq \varepsilon - \text{negl}(|x|)$.

We proceed to argue that in the execution of E , $E_{\mathcal{WI}}$ can only output a false witness with negligible probability. Suppose not, that is, for infinitely many n $E_{\mathcal{WI}}$ outputs a false witness with some noticeable probability $\varepsilon'(n)$. Then we will use this fact to contradict the collision resistance property of SIG' . We do so in the following two steps:

1. We construct an efficient cheating UA prover P_{UA}^* for $(P_{\text{UA}}, V_{\text{UA}})$ that convinces V_{UA} with probability $\text{poly}(\varepsilon(n))$.
2. We use the extractor E_{UA} from the weak argument of knowledge property of $(P_{\text{UA}}, V_{\text{UA}})$ together with this P_{UA}^* to build a collision-finder for SIG' .

Step 1: Constructing P_{UA}^* . P_{UA}^* internally emulates P^* and proceeds as follows.

- $P_{\text{UA}}^{*O}(\text{pp})$ runs $c_0 \leftarrow P^{*O}(x, \text{pp})$, samples $r \leftarrow \{0, 1\}^n$ and outputs $y = (c_0, r, \text{pp})$ as the adaptively chosen statement.
- $P_{\text{UA}}^{*O}(\text{pp})$ generates the first prover message p_1 as follows: $P_{\text{UA}}^{*O}(\text{pp})$ feeds r to P^{*O} , receives $c_1 \leftarrow P^{*O}(x, \text{pp}; r)$, and continues to emulate the interaction of P^{*O} with an honest V up to the end of Stage Two; let α be the partial transcript and $P^{*O}(x, \text{pp}; \alpha)$ be the “residual” prover. Then P_{UA}^* applies $E_{\mathcal{WI}}$ on $P^{*O}(x, \text{pp}; \alpha)$. If $E_{\mathcal{WI}}$ outputs a valid $(p_1, p_2, \tau_1, \tau_2) \in \mathbf{R}_{L_2}$, then P_{UA}^* outputs p_1 , otherwise, P_{UA}^* aborts.
- Upon receiving r' from V_{UA} , P_{UA}^* rewinds P^* until the point where it awaits the message r' , feeds r' to P^{*O} , and receives $c_2 \leftarrow P^{*O}(x, \text{pp}; r')$; let α' denote the partial transcript. Then P_{UA}^* applies $E_{\mathcal{WI}}$ on $P^{*O}(x, \text{pp}; \alpha')$. If $E_{\mathcal{WI}}$ outputs a valid $(p'_1, p'_2, \tau'_1, \tau'_2) \in \mathbf{R}_{L_2}$, then P_{UA}^* outputs p'_1 , otherwise, P_{UA}^* aborts.

Clearly by inspection, P_{UA}^* runs in expected polynomial time. Furthermore, we can make P_{UA}^* run in strict polynomial time by cutting it off after a certain polynomial time bound with only a small loss in its success probability. It follows by an identical argument to [10, 105] that P_{UA}^* convinces V_{UA} to accept with probability $\text{poly}(\varepsilon')$. Roughly, the argument consists of counting “good” oracles and verifier messages, i.e., those that will let the prover succeed with “high” probability (see Claim 4.2.1 in [10]), together with applying the binding property of the commitment scheme to show that the witnesses extracted by the two executions of E_{WT} have consistent first prover messages (i.e., $p_1 = p'_1$) except with negligible probability (See Lemma A.3 in [105]).

Step 2: Finding collision. We now use P_{UA}^* to break the collision resistant property of sig-com tree corresponding to SIG' , which contradicts Lemma 6. Let $1/p$ be a lower bound on the success probability of P_{UA}^* for some polynomial p . Let E_{UA} be the corresponding weak knowledge extractor for $(P_{\text{UA}}, V_{\text{UA}})$. Recall the weak argument of knowledge property guarantees that

$$\Pr[\text{pp}, O \leftarrow O(1^n); \omega, \nu \leftarrow \{0, 1\}^\infty; y \leftarrow P_{\text{UA}, \omega}^*(\text{pp}) : \exists w = w_1, \dots, w_t \in \mathbf{R}_{L_1}(y) \text{ s.t.} \\ \forall i \in [t], E_{\text{UA}, \nu}^{P_{\text{UA}, \omega}^*}(\text{pp}, y, i) = w_i] > \frac{1}{p'(n)},$$

where ω, ν denote the random tapes of P_{UA}^* and E_{UA} , respectively, p' is some polynomial, and the witness w is of the form $(\tau_0, d, l_\lambda, C, \{\vec{\beta}_i\}_{i \in [2^d]})$. To simplify the notation, let $\text{suc}(\text{pp}, O, \nu, \omega) = 1$ if the extraction successfully extracts a valid witness $w \in \mathbf{R}_{L_1}(y)$.

We construct a PPT adversary A that breaks the collision resistance property of sig-com tree corresponding to SIG' . A on input 1^n and vk with oracle access to a signing oracle $\text{Sign}_{\text{sk}}(\cdot)$ proceeds as follows.

- A uses vk and its signing oracle to emulate an \mathcal{O}^{SIG} oracle with $\text{pp} = \text{vk}$ and $\mathcal{O} = \text{Sign}_{\text{sk}}(\cdot)$.
- A samples $\omega, \tilde{\omega}$, and let $y \leftarrow P_{\text{UA},\omega}^{*O}(\text{pp}), \tilde{y} \leftarrow P_{\text{UA},\tilde{\omega}}^{*O}(\text{pp})$; recall from our definition of P_{UA}^* that $y = (c_0, r, \text{pp}), \tilde{y} = (c_0, \tilde{r}, \text{pp})$ will each contain the same c_0 and pp components, while r and \tilde{r} are selected independently uniformly at random.
- A samples $\nu, \tilde{\nu}$, and applies $E_{\text{UA},\nu}^{P^*O}(\text{pp}, y, \cdot)$ and $E_{\text{UA},\tilde{\nu}}^{P^*O}(\text{pp}, \tilde{y}, \cdot)$ to extract (part of) witnesses $w = (\tau_0, d, l_\lambda, C, \{\vec{\rho}_i\}_{i \in [2^d]})$ and $\tilde{w} = (\tilde{\tau}_0, \tilde{d}, \tilde{l}_\lambda, \tilde{C}, \{\vec{\rho}_i\}_{i \in [2^d]})$ as follows: (1) A first extracts (d, l_λ) and $(\tilde{d}, \tilde{l}_\lambda)$. A aborts if the extraction fails at any point. (Note that by the binding property of the commitment, $(d, l_\lambda) = (\tilde{d}, \tilde{l}_\lambda)$ except with negligible probability.) (2) Then A samples $i \leftarrow [2^d]$, and extracts $(C_i, \vec{\rho}_i)$ and $(\tilde{C}_i, \vec{\rho}_i)$
- If $C_i \neq \tilde{C}_i$, then A outputs $(\vec{\rho}_i, \vec{\rho}_i, i, l_\lambda)$ if $C_i = 0$, and $(\vec{\rho}_i, \vec{\rho}_i, i, l_\lambda)$ if $C_i = 1$.

We now show that A can break the collision resistance property with non-negligible probability. Note that A runs the knowledge extraction twice with respect to the same pp, \mathcal{O} but independent copies of (ν, ω) and $(\tilde{\nu}, \tilde{\omega})$. Recall that the extraction succeeds with probability at least $1/p'$. For at least $1/(2p')$ -fraction of (pp, \mathcal{O}) , it holds that $\Pr[\text{suc}(\text{pp}, \mathcal{O}, \nu, \omega) = 1 | (\text{pp}, \mathcal{O})] \geq 1/(2p')$. Therefore, with probability at least $(1/2p')^3$ over $(\text{pp}, \mathcal{O}, \nu, \omega, \tilde{\nu}, \tilde{\omega})$, both $\text{suc}(\text{pp}, \mathcal{O}, \nu, \omega) = \text{suc}(\text{pp}, \mathcal{O}, \tilde{\nu}, \tilde{\omega}) = 1$, i.e., both extractions invoked by A succeed.

Finally, we note that the independently drawn r and \tilde{r} are different with overwhelming probability, and so the Π and $\tilde{\Pi}$ underlying C and \tilde{C} will also be different with overwhelming probability. Since we used an error-correcting code ECC with constant min-distance, if $\Pi \neq \tilde{\Pi}$, then for a randomly chosen i , $C_i \neq \tilde{C}_i$ with constant probability c , meaning the two paths outputted by A

will have different leaf labels. Thus with probability $\geq c/(2p')^3$, A successfully outputs a pair of colliding paths with the same root but different leaf labels, breaking the collision resistance of sig-com trees corresponding to SIG' . \square

Finally, we apply the PRF transformation of [11] to the \mathcal{O}^{SIG} -oracle aided ZK protocol (P, V) constructed above to achieve \mathcal{O}^{SIG} -oracle aided resettable soundness. More precisely, we modify the public-coin verifier V to a “PRF-verifier” \tilde{V} that samples a seed s for a PRF f_s at the start of the protocol and then generates each verifier message by applying f_s to the current transcript. The proof in [11] shows how to transform a cheating resetting prover for this protocol into a stand-alone cheating prover breaking the argument of knowledge property, thus showing resettable-soundness from the argument of knowledge property. Further, the proof does so using the cheating prover only as a black box and thus the proof relativizes even to the setting when the prover is aided by an oracle. As a consequence we have the following theorem:

Theorem 10. *Let SIG' be a canonical sig-com scheme with SIG and Com being its underlying signature scheme and commitment scheme. Then there exists an \mathcal{O}^{SIG} -aided constant-round resettably-sound argument of knowledge (P, V) for NP ; additionally,*

1. P does not make any queries to its oracle;
2. (P, V) is (SIG', ℓ) -oracle-aided zero-knowledge for $\ell(n) = 2n$.

2.5 Resetably-sound Zero Knowledge in the Plain Model

Let SIG' be a canonical sig-com scheme with SIG and Com being its underlying signature scheme and commitment scheme. Let (P, V) be an \mathcal{O}^{SIG} -aided reset-

tably sound argument of knowledge for a language L with witness relation R_L , where P does not make any queries to its oracle. Consider the protocol (\tilde{P}, \tilde{V}) that on common input x , and auxiliary prover input w proceeds as follows.

1. Init: \tilde{V} runs $(\text{sk}, \text{vk}) \leftarrow \text{Gen}(1^n)$ and sends vk to \tilde{P} .
2. Signing Slot:
 - \tilde{P} generates $c = \text{Com}(0^{2n}; \tau)$, where τ is uniformly sampled, and sends c to \tilde{V} .
 - \tilde{V} replies with $\sigma = \text{Sign}_{\text{sk}}(c)$.
 - \tilde{P} aborts if σ is not a valid signature of c .
3. Body: Invoke the protocol $(P(w), V)(x, \text{pp})$ with $\text{pp} = \text{vk}$.

Lemma 11. *If (P, V) is $(\text{SIG}', 2n)$ -oracle-aided resettably-sound zero-knowledge for L with witness relation R_L , then (\tilde{P}, \tilde{V}) is a fixed-input resettably-sound zero-knowledge argument of knowledge for L with witness relation R_L .*

Note that here we only obtain a *fixed-input* resettably sound argument of knowledge (defined in Definition 6), but this can be transformed into a “full-fledged” resettably sound one by using the transformation in Claim 2, which thus establishes our main Theorem 1.

Before proving Lemma 11 formally, we provide a high-level sketch first. Completeness of (\tilde{P}, \tilde{V}) follows directly from the completeness of (P, V) . Resettably-soundness and the argument of knowledge property, roughly speaking, follow by turning the cheating prover for (\tilde{P}, \tilde{V}) into a cheating prover for (P, V) , and, when doing so, emulating all messages in the signature slot of (\tilde{P}, \tilde{V})

by forwarding them to the signature oracle in (P, V) . The zero-knowledge simulator proceeds by first honestly emulating the signature slot for the malicious verifier V^* , and if V^* provides an accepting signature, we next run the oracle-aided simulator, and appropriately rewind the malicious verifier during the signature slot to appropriately implement *some* valid sig-com oracle. The verifier may not always answer, but we can “keep rewinding” him, sending fresh commitments until he does. Roughly speaking, the key point is that if V^* did provide a valid signature during the first pass, then in expectation, by the hiding property of the commitment scheme, we only need a polynomial number of rewindings. This “almost” works: just as in [68], we need to take special care to deal with verifier’s that only provide valid signatures with very small probability. We proceed with a formal proof.

Proof. Completeness of (\tilde{P}, \tilde{V}) follows directly from the completeness of (P, V) since by assumption, P never makes any oracle queries.

To prove the fixed-input resettable-soundness of (\tilde{P}, \tilde{V}) , we show how to convert a malicious prover \tilde{P}^* for (\tilde{P}, \tilde{V}) into an *oracle-aided* malicious prover P^* for (P, V) that succeeds with the same probability. $P^{*O}(1^n, \text{pp})$ internally emulates an execution of \tilde{P}^* as follows:

- Upon invocation P^* feeds \tilde{P}^* the message pp (corresponding to the “Init message” of the protocol).
- Whenever \tilde{P}^* makes a signing slot query, that is, whenever it requests a signature on some message c from \tilde{V} , P^* forwards c to its oracle O , and relays the answer back to \tilde{P}^* as \tilde{V} ’s reply.
- All other messages are forwarded externally to the verifier, and the veri-

fier’s replies are relayed back.

It follows by inspection that P^* succeeds in convincing V (during a reset attack) with identically the same probability as \tilde{P}^* convinces \tilde{V} , since the view of \tilde{P}^* in the emulation by P^* is identical to its view in the execution with \tilde{V} .

By the same argument we have that (\tilde{P}, \tilde{V}) is an argument of knowledge: Let E be the extractor for (P, V) , and define the extractor \tilde{E} for (\tilde{P}, \tilde{V}) that given oracle access to \tilde{P}^* , proceeds as follows: \tilde{E} runs pp , $O \leftarrow O$, and then \tilde{E} emulates the execution of E given oracle access to P^* described above, while 1) internally emulating all oracle queries by P^* (using O) and 2) externally querying (and relaying back the answer) \tilde{P}^* on all queries made by E to P^* . Since \tilde{P}^* succeeds in convincing \tilde{V} with identically the same probability as P^* convinces V , it follows by the argument of knowledge property of (P, V) that (\tilde{P}, \tilde{V}) also is an argument of knowledge.

Let us turn to zero-knowledge. Consider some malicious (w.l.o.g. deterministic) verifier \tilde{V}^* for (\tilde{P}, \tilde{V}) of size $T_{\tilde{V}^*}$. We construct a simulator \tilde{S} for \tilde{V}^* . Roughly speaking, \tilde{S} starts by simulating (\tilde{P}, \tilde{V}^*) honestly up to the end of the Signing Slot, and if \tilde{P} does not abort, \tilde{S} continue to simulate the view of \tilde{V}^* in the Body part by 1) viewing the “residual” \tilde{V}^* as a malicious V^* for (P, V) , 2) preparing a valid $(\text{SIG}', 2n)$ oracle O' (by rewinding \tilde{V}^* at the Signing Slot in the spirit of Goldreich-Kahan [68]), and 3) invoking the simulator S for V^* with oracle O' .

More precisely, \tilde{S} first receives vk from \tilde{V}^* , generates and sends to \tilde{V}^* an honest commitment $c = \text{Com}(0^{2n}; \tau)$ with uniform τ , and then receives back a signature σ from \tilde{V}^* . If σ is not a valid signature of c , then the simulation halts immediately and outputs the transcript upto that point. Otherwise, let V^* be the

residual \tilde{V}^* at the end of the Signing Slot (which is a malicious verifier for (P, V)), and construct an oracle \mathcal{O}' as follows.

- \tilde{S} repetitively queries \tilde{V}^* at the Signing Slot with fresh commitments $\text{Com}(0^{2n}; \tau)$ until it collects $2n$ valid signatures. Let t be the number of queries \tilde{S} makes.
- Define \mathcal{O}' that outputs $\text{pp} = \text{vk}$, and an oracle \mathcal{O} that on input a message $m \in \{0, 1\}^{2n}$, proceeds as follows: \mathcal{O} repetitively queries \tilde{V}^* at the Signing Slot with fresh commitments $\text{Com}(m; \tau)$ for at most t times. If \tilde{V}^* ever replies a valid signature σ for $\text{Com}(m, \tau)$, then \mathcal{O} outputs (σ, τ) . Otherwise, \mathcal{O} returns \perp .

If $t \geq 2^{n/2}$, then \tilde{S} aborts. Otherwise, \tilde{S} invokes the simulator S for V^* with oracle \mathcal{O}' , while emulating the oracle for S during its execution, and outputs the view of V^* (which is also a view of \tilde{V}^*) generated by S at the end.

To analyze \tilde{S} , we introduce some notation. Let $p(m)$ be the probability that \tilde{V}^* on query a random commitment $c = \text{Com}(m, \tau)$ of $m \in \{0, 1\}^{2n}$ at the Signing Slot, returns a valid signature of c . Let $p = p(0^{2n})$.

We first show that \tilde{S} runs in expected polynomial time. To start, note that \tilde{S} aborts at the end of the Signature Slot with probability $1 - p$, and in this case, \tilde{S} runs in polynomial time. With probability p , \tilde{S} continues to invoke a strictly polynomial-time simulator S for the residual V^* , which has size bounded by $T_{\tilde{V}^*}$. Thus, S runs in some $T = \text{poly}(T_{\tilde{V}^*})$ time and makes at most T queries to its oracle \mathcal{O} , which in turn runs in time $t \cdot \text{poly}(n)$ to answer each query. Also note that \tilde{S} runs in time at most 2^n , since \tilde{S} aborts when $t \geq 2^{n/2}$. Now, we claim that $t \leq 10n/p$ with probability at least $1 - 2^{-n}$, and thus the expected running time of

\tilde{S} is at most

$$(1 - p) \cdot \text{poly}(n) + p \cdot T \cdot (10n/p) \cdot \text{poly}(n) + 2^{-n} \cdot 2^n \leq \text{poly}(T_{\tilde{V}^*}, n).$$

To see that $t \leq 10n/p$ with overwhelming probability, let $X_1, \dots, X_{10n/p}$ be i.i.d. indicator variables on the event that \tilde{V}^* returns a valid signature for a random $\text{Com}(0^{2n}; \tau)$, and note that $t \leq 10n/p$ implies $\sum_i X_i \leq 2n$, which by a standard Chernoff bound, can only happen with probability at most 2^{-n} .

Finally, we argue indistinguishability. First, the computational hiding property of Com implies that there exists some negligible $\nu(\cdot)$ such that $|p(m) - p| \leq \nu(n)$ for every $m \in \{0, 1\}^{2n}$. Now we consider two cases. If $p \leq 2\nu$, then the indistinguishability trivially holds since the interaction aborts at the end of the Signature Slot (in this case, the view is perfectly simulated) with all but negligible probability. On the other hand, if $p \geq 2\nu$, we show that \mathcal{O}' generated by \tilde{S} is a valid $(\text{SIG}', 2n)$ oracle for SIG' with overwhelming probability, and thus the indistinguishability of \tilde{S} follows by the indistinguishability of S .

To see that \mathcal{O}' is a valid $(\text{SIG}', 2n)$ oracle for SIG' with overwhelming probability, note again by a Chernoff bound that $n/p \leq t \leq 2^{n/2}$ with probability at least $1 - 2^{-\Omega(n)}$. In this case, for every $m \in \{0, 1\}^{2n}$, $p(m) \geq p - \nu \geq p/2$ implies that $t \geq n/2p(m)$, and thus $\mathcal{O}(m)$ learns a valid signature of $\text{Com}(m; \tau)$ from \tilde{V}^* with probability at least $1 - 2^{-\Omega(n)}$. \square

2.6 PCP-Free RSZK

In this section, we provide an alternate construction of RSZK-AoK from one-way functions, without relying on the use of universal arguments and PCPs.

That is, we reprove the following theorem:

Theorem 12. *Assume the existence of one-way functions. Then there exists a constant-round resettably-sound zero knowledge argument of knowledge for all of NP.*

Note that our construction, as described below, makes use of CRHs, but these can be replaced with sig-coms as in the protocol in Figure 2.1. That is, wherever hashes are used, we can replace them with sig-coms, and add a rewindable signature slot at the start of the protocol to allow the simulator to compute these sig-coms. Additionally, in the protocol described below, the honest prover will be able to complete the protocol without hashing. These features will allow us to dispense with CRHs and achieve a protocol based on only OWFs. The formal proof for removing CRHs is exactly analogous to the approach taken in Sections 2.4 and 2.5, and we omit it here.

Our construction will make crucial use of *verifiable* Turing machine steps. To this end, we define the following:

Definition 20 (TM description). *Let M be a Turing Machine with a single read/write input/work tape, and a read-only auxiliary input tape holding auxiliary input string z , with $|z| = \text{poly}(n)$, where $n = |x|$, the initial input on the input tape. Further, assume that both tapes have alphabet $\{0, 1\}$, and the input/work tape is of size 2^n (sufficient for poly-time TMs) while the auxiliary tape is of fixed size $= |z|$.*

Let h be a hash function that hashes strings of length $2n$ to length n . We say a string $m = (q_0, Q, F, \delta, d_{aux}, r_{aux})$ is a description of M w.r.t. h if q_0, Q, F and δ are binary representations of M 's start state, state set, final state set and transition function respectively, while d_{aux}, r_{aux} are the depth and root of a hash tree for z . Note that, for sufficiently large n , $|m| \leq 2n$. We pad m so that it has length exactly $2n$.

Note: We require the hash trees of TM main-tapes to have a particular form. They should always have depth exactly n . The tape corresponding to the input tape should contain exactly the input to the TM (i.e., no following blank spaces). If this results in a partially full hash tree, the empty subtrees should be represented by placing hashes to \perp at their roots, with no children. When the TM accesses a previously unexplored space on the right of the tape, if the tree is partially full, the tape should be extended by one, adding a "blank" character to the newly explored tape space, and adjusting the \perp symbols to reflect the updated tape length.

Definition 21 (TM configuration). *Let n be a parameter and h be a hash function that hashes strings of length $2n$ to length n . We say that $c = (q, r, i, i')$ is a valid TM configuration w.r.t. h if q is a TM state, r is the root of a hash tree of a main tape, and i and i' are the positions of the main tape head and auxiliary tape head respectively, with the leftmost position being 0 and counting upwards to the right. For sufficiently large n , $|c| \leq 2n$. We pad c so it has length exactly $2n$.*

Returning to our construction of RSZK-AoK without universal arguments, our strategy will be as outlined in Section 2.1.2. Recall that in Barak's protocol, universal arguments (and thus PCPs) are used to prove that a commitment c is a commitment to a hash of a program M such that $M(c) = r$, and that it is only the simulator that needs to prove the above statement. Rather than providing a universal argument as in [10], we let the simulator prove this statement *piecemeal*, that is, by getting every step of M 's computation on c signed by the verifier. We accomplish this by making use of a rewindable *verified computation slot*.

More precisely, the actual protocol proceeds in three stages:

Stage One This stage consists of setup, the prover’s commitment, and the verifier’s challenge. The verifier generates a key-pair vk, sk for a strong, length- n signature scheme, and sends vk to the prover, together with a collision resistant hash function h , and setup information for a commitment scheme. The prover sends a commitment c_0 to a *hashed-down* representation of a Turing Machine M (including the auxiliary tape), and the verifier responds with a challenge r .

Stage Two This stage consists of a verified computation slot, in which the prover is allowed to receive signatures w.r.t. sk on Turing machine states from the verifier. The prover sends one of

1. A tuple $(start, start, M)$, where $start$ is a starting TM configuration for the machine M committed to in Stage One. (A TM configuration includes the current state and main tape, *hashed down*. The description of M includes the description of transition function, and the contents of the auxiliary tape, hashed down. See Definition 20 for the exact form we use).
2. A pair of tuples $(start_1, current_1, M_1)$ and $(start_2, current_2, M_2)$, where $start_1, start_2, current_1, current_2$ are TM configurations, M_1, M_2 are TM descriptions, with $M_1 = M_2$ and $start_1 = start_2$ (again, both TM configurations and descriptions are hashed-down and has fixed bounded length). Additionally, the prover must show that the tuple $(start_1, current_1, M_1)$ has been signed using sk , and that $current_2$ has been generated from $current_1$ by a single step of computation of machine M_1 .

The verifier is to reply with a signature, in the first case, of the tuple $(start, start, M)$, and in the second case, to $(start_2, current_2, M)$.

In fact, the above description is insufficient: we also need some form of blinding to prevent a cheating verifier from discriminating between different messages sent in this stage. Thus the prover, rather than sending the above tuples in the clear, instead sends a pair of *commitments* c_1, c_2 in this stage, and additionally proves the OR of the following statements using a rsWI-AOK:

- (a) $x \in L$
- (b) c_2 is a commitment to a tuple of the form $(start, start, M)$
- (c) c_1 is a commitment to a sig-com of a tuple $(start_1, current_1, M_1)$ w.r.t vk , c_2 is a commitment to a tuple $(start_2, current_2, M_2)$, $start_1 = start_2$, $M_1 = M_2$, and $current_2$ was generated from $current_1$ by a single step of M_1

If the proof is convincing, V is to respond with a signature σ on c_2 . Note that an honest prover can give dummy commitments to 0^n in c_1 and c_2 , and pass through the proof using the $x \in L$ condition. However, a simulator can make use of the latter two conditions, and can rewind the proof and send different pairs c_1 and c_2 to sign the whole computation path of a TM M (using the first condition to sign the starting configuration, and the second condition to sign successive configurations in the computation path.)

Stage Three The prover sends the verifier a commitment c_3 , and gives a rsWI-AoK to the verifier that either

1. $x \in L$, or
2. c_3 is a commitment to a sig-com for a tuple $(start, final, M)$, such that M is the same machine committed to in c_0 , $start$ is a starting configura-

tion of M with input tape = c_0 , and that $final$ is a halting configuration of M , where the first n bits of the main tape = r .

If the proof is convincing, the verifier halts and accepts.

Roughly speaking, the “slot” in Stage Two allows the simulator (or a resetting prover) to get a signature to a commitment of $(start, final, M)$, corresponding to a completed computation by a TM. This is achieved by using condition (b) of Stage Two to get a sig-com to $(start, start, M)$ where $start$ is the initial configuration of M on input c_0 , and, by rewinding and using condition (c) of Stage Two, get sig-coms to all intermediate configurations $(start, current_i, M)$ of M .

By committing to the verifier’s code V^* as M , the simulator can get a signature on $(start, final, V^*)$ where $final$ is the terminating configuration of the computation of $V^*(c_0)$. The simulator can then use this signature as a trapdoor in Stage Three to convince the verifier that $M(c_0) = r$ where M is the program committed to in c_0 , since if $M = V^*$, then $V^*(c_0)$ is indeed equal to r .

The full description of the protocol is available in Figure 2.4. Note that the protocol makes use of collision-resistant hash function trees, but that we can replace them, as described earlier, using sig-com schemes.

Completeness of the protocol follows immediately, since an honest prover can simply prove $x \in L$ in each of Stages Two and Three, and thus convince the honest verifier.

We now prove the argument of knowledge property, which also implies soundness of the protocol.

Common Input: An instance $x \in \{0, 1\}^n$ of a language $L \in \text{NP}$ with witness relation \mathbf{R}_L .

Auxiliary input to P : A witness w such that $(x, w) \in \mathbf{R}_L$.

Primitives Used:

- SIG a strong length- n signature scheme.
- Com a commitment scheme that produces commitments of length n^2 for messages of length n .
- A collision-resistant hash-function family, \mathcal{HF} that hashes messages of length $2n$ to outputs of length n .
- A pseudorandom function family \mathcal{F}

Stage One (Trapdoor):

V_1 : Generate $(\text{sk}, \text{vk}) \leftarrow \text{Gen}(1^n)$, $h \leftarrow \mathcal{HF}(1^n)$, and send vk, h to P .
Choose $f \leftarrow \mathcal{F}$

P_1 : Send $c_0 = \text{Com}(0^{2n}, \tau_0)$, for uniformly selected τ_0 .

V_2 : Generate $r \in \{0, 1\}^n$ by applying f to P_1 , and send r to P

Stage Two (Verified Computation slot):

P_2 : Send $c_1 = \text{Com}(0^{6n}, \tau_1)$ and $c_2 = \text{Com}(0^{6n}, \tau_2)$, for uniformly selected τ_1 and τ_2 .

$P \Leftrightarrow V$: A resettably-sound WI-AOK $\langle P_{\text{WI}}, V_{\text{WI}} \rangle$ proving the OR of the following statements:

- a) $\exists w \in \{0, 1\}^{\text{poly}(|x|)}$ s.t. $(x, w) \in \mathbf{R}_L$.
- b) $\exists \langle m_2, \tau_2 \rangle$ such that $c_2 = \text{Com}(m_2, \tau_2)$, and $m_2 = (s, s, M)$, where s is a *TM* configuration w.r.t. h , and M is a *TM* description w.r.t. h (see Def. 20).
- c) \exists
s.t.
 $\langle \sigma_1, c'_1, m_1, m_2, \tau_1, \tau'_1, \tau_2, \rho_1, \rho_2, \rho_3, \rho_4 \rangle$
 $\langle \langle c_1, c_2, \text{vk}, h \rangle, \langle \sigma_1, c'_1, m_1, m_2, \tau_1, \tau'_1, \tau_2, \rho_1, \rho_2, \rho_3, \rho_4 \rangle \rangle \in \mathbf{R}_{L_3}$ (defined in Fig. 2.6)

V_3 : If the above proof is convincing, send $\sigma = \text{Sign}_{\text{sk}}(c_2)$, otherwise abort.

(Continued in Figure 2.5).

Figure 2.4: A PCP-free resettably-sound ZK argument of knowledge.

Stage Three: (Main Proof)

P_3 : If the signature σ is invalid, abort. Otherwise, send $c_3 = \text{Com}(0^{6n}, \tau_3)$ for uniformly selected τ_3 .

$P \Leftrightarrow V$: A resettably-sound WI-AOK $\langle P_{\text{WI}}, V_{\text{WI}} \rangle$ proving the OR of the following statements:

1. $\exists w \in \{0, 1\}^{\text{poly}(|x|)}$ s.t. $(x, w) \in \mathbf{R}_L$.
2. $\exists \langle m_0, \sigma_3, c'_3, m_3, \tau_0, \tau_3, \tau'_3, \vec{\rho}_s, \vec{\rho}_f \rangle$ s.t.
 $(\langle c_0, c_3, r, vk, h \rangle, \langle m_0, \sigma_3, c'_3, m_3, \tau_0, \tau_3, \tau'_3, \vec{\rho}_s, \vec{\rho}_f \rangle) \in \mathbf{R}_{L_4}$ (defined in Fig. 2.8).

V accepts if the above proof is convincing, otherwise aborts.

Figure 2.5: A PCP-free resettably-sound ZK argument of knowledge, Contd.

2.6.1 Proof of Argument of Knowledge property

To prove the argument of knowledge property for this protocol, our approach will be analogous to the one we used for the previous, PCP-based, resettably-sound ZK-AOKs. That is, we will first consider a public-coin oracle-aided version of the protocol, and show that, in this oracle model, the protocol is a public-coin oracle-aided argument of knowledge. Using the PRF transformation of [11], we can convert this public-coin protocol into an oracle-aided resettably sound argument of knowledge. Then we will eliminate the oracle by adding a rewindable “slot” that provides the same functionality as the oracle, and show how to convert any attacking prover in the plain model into an attacking prover for the oracle model having similar probability of success. Just as in Section 2.5, this will be enough to show that the protocol in the plain model is also a resettably sound argument of knowledge.

We begin by defining the oracles used in the oracle model. It is essentially a reusable, concurrent, oracle version of the verified computation slot of the protocol in Fig. 2.4. However, note that in the description below, the oracle does

Relation 3: Let SIG a strong length- n signature scheme, Com a commitment scheme that produces commitments of length n^2 for messages of length n , and $\mathcal{HF}(1^n)$ a collision-resistant hash-function family. We say that $\langle c_1, c_2, \text{vk}, h \rangle \in L_3$ if $\exists \langle \sigma_1, c'_1, m_1, m_2, \tau_1, \tau'_1, \tau_2, \rho_1, \rho_2, \rho_3, \rho_4 \rangle$ showing, roughly, that c_1 contains a signed partial TM computation, and that c_2 contains that same computation, unsigned, moved forward one more step. The details are as follows:

- c_1 is a commitment to a sig-com for $m_1 = (\text{start}, \text{current}_1, M)$ with $c_1 = \text{Com}((c'_1, \sigma_1), \tau_1)$, $\text{Ver}_{\text{vk}}(\sigma_1, c'_1) = 1$, $c'_1 = \text{Com}(m_1, \tau'_1)$ and further, each of start and current_1 is a TM configuration w.r.t h , while M is a TM description with respect to h .
- c_2 is a (regular) commitment to $m_2 = (\text{start}, \text{current}_2, M)$ using τ_2 as randomness ($c_2 = \text{Com}(m_2, \tau_2)$), where each of start and current_2 is a TM configuration w.r.t. h , M is a TM description with respect to h , and start and M are identical to the corresponding elements in m_1 .
- ρ_1, ρ_2, ρ_3 and ρ_4 are hash tree paths that show that $\text{current}_2 = (q_2, r_2, i_2, i'_2)$ is the configuration derived by running a single step of M from $\text{current}_1 = (q_1, r_1, i_1, i'_1)$. That is,
 - ρ_1 and ρ_2 reveal the bit under the main tape head at position i_1 , in the hash trees rooted at r_1 and r_2 respectively (call these bits b_1 and b_2)
 - ρ_3 reveals the bit under the auxiliary tape head at position i'_1 , in the hash tree rooted at r_{aux} , corresponding to the i'_1 th bit of z (call this bit b_z).
 - The two configurations are consistent with M 's transition function δ . That is, $\delta(q_1, b_1, b_z) = (q_2, b_2, \text{dir}, \text{dir}')$, where dir and dir' correspond to the differences between (i_1, i_2) and (i'_1, i'_2) respectively. (e.g., if $\text{dir} = L$, corresponding to moving the main tape head left, then i_2 should be equal to $i_1 - 1$.)
 - (Contd. in Figure 2.7)

We let \mathbf{R}_{L_3} be the witness relation corresponding to L_3 .

Figure 2.6: Relations used in the PCP-free ZK protocol

Relation 3: (Contd. from Figure 2.6)

- – ρ_1 and ρ_2 also show that, except for the bit at position i_1 , the main tape in configurations $current_1$ and $current_2$ is identical. For this, we recall that a hash tree path for a leaf i must contain all nodes along the path from the root to leaf i , along with the *siblings* of each of those nodes. Notice that if each of these sibling nodes is identically valued in ρ_1 and ρ_2 , then this implies that the subtrees underlying those nodes are also equal (we use the word “implies” loosely: this only holds if we assume the collision-resistance of h is unviolated). We thus see that to check equality of main-tapes between c_1 and c_2 , it is sufficient to check that the sibling nodes of ρ_1 and ρ_2 are equal-valued at every level.
- In the special case for which the main-tape head moved right into a previously unexplored space on the main tape, the previous condition is modified so that only that the *left* siblings of the paths ρ_1 and ρ_2 match. But additionally all the right siblings along the path in ρ_1 must be equal to the hash of \perp , showing that the tape is empty to the right of position i_1 ^a. Additionally, path ρ_4 opens up position $i_1 + 1$, showing that a new blank space was correctly generated in that location, and that the remainder of the tree on the right is empty (or in other words, all the right siblings in ρ_4 contain the hash of \perp). Finally, ρ_2 and ρ_4 must be consistent, that is, all nodes in the tree that appear in both ρ_2 and ρ_4 (including both sibling nodes and nodes along the path) must have identical values in each of ρ_2 and ρ_4 .

We let \mathbf{R}_{L_3} be the witness relation corresponding to L_3 .

^aWhen replacing hashes with sig-coms, we note that there may be multiple different valid sig-coms for \perp , and they may be hard to recognize. We thus require that anytime a hash tree path needs to show that a particular node contains a sig-com of \perp , auxiliary information showing that the sig-com contains \perp (i.e. the commitment underlying the signature, together with the opening of the commitment to \perp) is to be appended to the end of the hash-tree path.

Figure 2.7: Relations used in the PCP-free ZK protocol (contd.)

Relation 4: Let L_3 be described as above, with respect to schemes SIG , Com and \mathcal{HF} . We say that $\langle c_0, c_3, r, vk, h \rangle \in L_4$ if $\exists \langle m_0, \sigma_3, c'_3, m_3, \tau_0, \tau_3, \tau'_3, \vec{\rho}_s, \vec{\rho}_f \rangle$ showing, roughly, that c_3 contains a signed, complete computation of a TM M , which is the same machine contained in M , and that this computation ends with the value r on the main tape. In more detail:

- $c_0 = \text{Com}(m_0, \tau_0)$, where M is a TM description with respect to h .
- c_3 is a commitment to a sig-com for m_3 , with $c_3 = \text{Com}(\sigma_3, \tau_3)$, $\text{Ver}_{vk}(\sigma_3, c'_3) = 1$, while $m_3 = (\text{start}, \text{final}, M)$ is a tuple consisting of TM configurations w.r.t. h , followed by a TM description M which is identical to that in m_0 .
- We expand $M = (q_0, Q, F, \delta, d_{aux}, r_{aux})$, $\text{start} = (q_s, r_s, i_s, i'_s)$, and $\text{final} = (q_f, r_f, i_f, i'_f)$.
- start is a starting configuration of M , with input c_0 . That is, $q_s = q_0$, $i_s = i'_s = 0$, and $\vec{\rho}_s$ reveals that the input tape of start consists of exactly c_0 , with the remainder of the tree being empty (consists of \perp).
- final is a final configuration of M , with final output $= r$. More precisely, we require that $q_f \in F$, and that $\vec{\rho}_f$ consist of n hash tree paths showing that the first n bits of final 's input tape match r . (All bits after the first n bits are ignored)

We let \mathbf{R}_{L_4} be the witness relation corresponding to L_4 .

Figure 2.8: Relations used in the PCP-free ZK protocol (contd.)

not use a PRF to produce its challenges (as the rsWI-AoK in the protocol of Fig. 2.4 would). Rather, it simulates a truly random function, by sending truly random challenges, and storing and resending the challenges on repeated queries. However, we will subsequently covert the oracle to use a PRF to generate its challenges, so that its functionality matches up with the verified computation slot in Stage Two of Fig. 2.4.

Definition 22 (Verified Computation Oracle). *Let $x \in \{0, 1\}^*$ be a statement, and L be an NP language with witness relation R_L . Let SIG be a strong signature scheme, Com a commitment scheme and \mathcal{HF} a collision-resistant hash function family. Let $\langle P_{WI}, V_{WI} \rangle$ be a 3-round WI-AOK.*

We say O is a Verified Computation Oracle (VCO) for the above primitives if, on input x , O generates a public parameter pp , and a corresponding oracle O as follows:

- $O(x)$ generates $(\text{sk}, \text{vk}) \leftarrow \text{Gen}(1^n)$, $h \leftarrow \mathcal{HF}(1^n)$, and setup for the commitment scheme to be used for the WI-AOK proofs (also the first verifier message), where $n = |x|$, and releases $\text{pp} = (\text{vk}, h, \text{setup})$.
- After generating the parameters, the oracle signs messages as follows:
 - (i) $O(c_1, c_2, p_1)$ checks if c_1 and c_2 are commitments, and p_1 is the first prover message for a WI-AOK proving the OR of the following conditions:
 - a) $x \in L$, or
 - b) $\exists \langle m_2, \tau_2 \rangle$ such that $c_2 = \text{Com}(m_2, \tau_2)$, and $m_2 = (\text{start}, \text{start}, M)$, where M is a TM description w.r.t. h , and start is a starting TM configuration for M w.r.t. h (see Defs. 20 and 21).
 - c) $(c_1, c_2, \text{vk}, h) \in L_3$ (See Fig 2.6). (Roughly, this is true if c_1 is a signed commitment to a partial TM computation $(\text{start}, \text{current}_1, M)$, and c_2 is commitment to $(\text{start}, \text{current}_2, M)$ where current_2 is derived from current_1 by one step of computation of M .)

If so, and if (c_1, c_2, p_1) has been queried before, O checks its table, and resends the response from the previous query of (c_1, c_2, p_1) . If (c_1, c_2, p_1) is being queried for the first time, O returns $r \leftarrow \{0, 1\}^n$, a freshly drawn verifier challenge, and stores r together with (c_1, c_2, p_1) in its table.

- (ii) $O(c_1, c_2, p_1, r, p_2)$ returns $\sigma = \text{Sign}_{\text{sk}}(c_2)$ if r was sent as a response to a previous query on (c_1, c_2, p_1) , and $(\text{setup}, p_1, r, p_2)$ together constitute an accepting proof for the aforementioned WI-AOK.

Note that proofs can be sent to the oracle in a fully concurrent manner: the

prover can send multiple queries for the same (c_1, c_2, p_1) , or start a new interaction with (c'_1, c'_2, p'_1) before sending the second message for a previous interaction. However, note that if the prover repeats a query (c_1, c_2, p_1) , then the oracle replies with the same challenge string r .

Given this definition of verified computation oracles, we now present the modified version of the protocol for the oracle model, in Fig 2.9. Notice that the protocol omits Stage Two, instead relying on the VCO for the equivalent functionality. Also, the rsWI-AOK in Stage Three has been replaced with a WI-AOK sending truly random challenges.

Lemma 13. *The VCO-aided public-coin protocol given in Fig 2.9 is an oracle-aided argument of knowledge.*

In fact, the protocol in Fig 2.9 has the additional property that an honest prover *never needs to use the oracle*. This fact is important when we return from the oracle model to the plain model.

We show this lemma in a manner similar to Theorem 9, which itself is adapted from [10] and the proof of Proposition 4.2 in [105]. The proof appears in Sections 2.6.1 and 2.6.1.

Description of Extractor for the Oracle-aided Scheme

The strategy for the extractor E is to rewind the proof in Stage Three, together with all the proofs in the oracle queries, and to extract the witnesses used in each of these proofs. E simply attempts to extract a witness for $x \in L$ from the above proofs with non-negligible probability. We argue that this extraction

Common Input: An instance $x \in \{0, 1\}^n$ of a language $L \in \text{NP}$ with witness relation \mathbf{R}_L .

Auxiliary input to P : A witness w such that $(x, w) \in \mathbf{R}_L$.

Primitives Used:

- SIG a strong length- n signature scheme.
- Com a commitment scheme that produces commitments of length n^2 for messages of length n .
- A collision-resistant hash-function family, \mathcal{HF} that hashes messages of length $2n$ to outputs of length n .
- A pseudorandom function family \mathcal{F}
- A Verified Computation Oracle O defined relative to SIG, Com, \mathcal{HF} and \mathcal{F}
- A 3-round public-coin WI-AOK

Set Up: Run $(\text{pp}, O) \leftarrow O(x)$, add pp to common input for P and V . Further, allow P oracle access to O .

Stage One (Trapdoor):

P_1 : Send $c_0 = \text{Com}(0^{2n}, \tau_0)$, for uniformly selected τ_0 .

V_1 : Send $r \xrightarrow{\$} \{0, 1\}^n$ to P

Stage Three: (Main Proof)

P_3 : Send $c_3 = \text{Com}(0^n, \tau_3)$ for uniformly selected τ_3 .

$P \Leftrightarrow V$: A public coin WI-AOK $\langle P_{\text{WI}}, V_{\text{WI}} \rangle$ proving the OR of the following statements:

1. $\exists w \in \{0, 1\}^{\text{poly}(|x|)}$ s.t. $(x, w) \in \mathbf{R}_L$.
2. $\exists \langle m_0, \sigma_3, c'_3, m_3, \tau_0, \tau_3, \tau'_3, \vec{\rho}_s, \vec{\rho}_f \rangle$ s.t.
 $(\langle c_0, c_3, r, vk, h \rangle, \langle m_0, \sigma_3, c'_3, m_3, \tau_0, \tau_3, \tau'_3, \vec{\rho}_s, \vec{\rho}_f \rangle) \in \mathbf{R}_{L_4}$ (defined in Fig. 2.8).

Figure 2.9: A PCP-free public-coin VCO-aided argument of knowledge.

must succeed in extracting a witness for $x \in L$ with non-negligible probability. If not, then we show how to use the cheating prover to break one of the commitment scheme the signature scheme, or the CRH. To do so, we simply look at the witnesses extracted by E : either these witnesses will immediately allow us to break one of the aforementioned schemes, or else, the prover must indeed have committed to a machine M in Stage One such that $M(c_0) = r$, and the witnesses will recover a step-by-step deterministic computation path showing $M(c_0) = r$. Then, we will simply rewind the cheating prover back to Stage One, and, keeping c_0 the same, send a different challenge r' and repeat the extraction process. It is now information-theoretically impossible that $M(c_0) = r'$, since M and c_0 are fixed in both interactions. Thus, by extracting the witnesses from the second run, we must have that P^* must break one of the commitment scheme, the signature scheme, or the CRH.

Let P^* be a VCO-aided prover that succeeds in convincing the honest verifier V to accept on input x with probability p , where p is taken over the randomness of P^* , V and O . Further, let $t(n)$ be a bound on the running time of P^* . Then, $t(n)$ is also a bound on the number of oracle queries made by P^* .

We now describe the running of the extractor E as follows:

1. E simulates a single, complete interaction of P^* with V on common input x . That is, it generates $(pp, O) \leftarrow O$, and sends it as common input to each player, and then runs the remainder of the oracle, simulating O for P^* .
2. If P^* fails to convince V , E halts and output \perp . If P^* succeeds in convincing V , then E extracts the witness from the public-coin WI-AOK in Stage

Three.

3. Further, E also extracts a witness from every successful WI-AOK sent to the VCO by P^* over the course of the original interaction. The procedure is as follows¹¹: For each instance (c_1, c_2) and first prover message p_1 such that (c_1, c_2, p_1) is queried to the VCO by P^* :

- If P^* never completes a convincing proof to the VCO for this query, E does no extra work.
- If P^* succeeds in this proof, E rewinds to the first time (c_1, c_2, p_1) was called to the oracle, and responds to the oracle query with a different random challenge r' . E then continues the simulation from this point, using fresh randomness for the remainder of the interaction.
- E repeats the previous step, sending different random challenges r' and using fresh randomness for the remainder of the execution, until P^* sends a second successful proof for instance (c_1, c_2) .
- If the challenges r' used by the oracle in the 2 runs are the same (which happens with negligible probability), E aborts. Otherwise, E combines the two responses to extract the witness for this proof.

4. If any of the witnesses extracted in the previous two steps is a w such that $(x, w) \in R_L$, then the extractor outputs w . Else, it fails and outputs \perp .

We will now argue that the above procedure runs in expected polynomial time.

- The first step, a single execution of P^* with V , takes time at most $t(n)$.

¹¹We note that, alternatively to the procedure described above, we could directly apply the concurrent extractor for the WI-AoK, see [93]. We instead include the details for our extractor above, both for clarity, and because it is simpler than the extractor in [93]

- Each of the extractions in step 2 and step 3 also take expected polynomial time $t(n)$ each. To see this, for any proof sent by P^* , let p be the probability of the P^* succeeding in the proof in the first interaction, conditioned on the transcript upto that point. Then, with probability $1-p$, the proof fails in the first interaction, and the extractor must rewind the proof 0. Conversely, with probability p , the proof succeeds, and the extractor rewinds $1/p$ times in expectation, with each rewinding taking time at most $t(n)$. Overall, the expected running time of E for each proof sent by P^* is $(p \cdot 1/p) \cdot t(n) = t(n)$. Since there can be at most $t(n)$ total executions of steps 2 and 3 combined (since P^* can make at most $t(n)$ oracle queries), the total expected running time for extractions is bounded by $t(n)^2$

Thus the overall expected running time for E is $t(n) + t(n)^2$, which is expected polynomial time.

It remains to show that E successfully extracts a witness w for $(x, w) \in \mathbf{R}_L$ with probability $p - \text{negl}(n)$. Note that since each of the extractions fails with only negligible probability, and there are only polynomially many extractions performed by E , we do extract all of the witnesses for the proofs in Stage Three and the oracle queries with probability $p - \text{negl}(n)$. However, we may extract *false* witnesses in each one of these proofs, and hence output \perp instead of w . (By false witnesses, we mean witnesses for the alternate conditions allowed for in each of these proofs, besides the option of proving that $x \in L$). The lemma proved in the following section will show that we cannot extract all false witnesses except with negligible probability.

Correctness of the Extractor

Here we show that the extractor cannot extract “all false witnesses” (i.e., all witnesses that use conditions besides $x \in L$) except with negligible probability.

We sketch our approach as follows: suppose that an execution of E extracts all false witnesses with non-negligible probability. Then, in particular, it must extract a false witness in Stage Three, namely, a signed final state of a Turing machine M committed to in Stage One. Assuming P^* does not break the security of the commitment scheme or the signature scheme, it must have received this signed final state by providing a sequence of convincing proofs to the VCO showing the computation of M . By examining the Stage Two witnesses extracted by E , we can reconstruct the deterministic computation path showing that $M(c_0) = r$.

Once we have this signed computation path in hand, we rewind the extractor’s interaction with P^* , and have the extractor send a different challenge r' as V_1 . With non-negligible probability, P^* will succeed in this execution as well, and further, E will extract all-false witnesses. By the same procedure, extracted witnesses will provide a second deterministic computation path, showing this time that $M(c_0) = r'$. However, we now have two paths showing that the same machine M running deterministically on the same input c_0 produces two different outputs r and r' , which means the prover must cheat somewhere along the way. Tracking the point where the computation diverges, we can locate a point where P^* breaks the security of the commitment scheme, the CRH, or the signature scheme.

We formalize this sketch in the following lemma:

Lemma 14. *Let P^* be a malicious prover for the VCO-aided protocol in Fig. 2.9, such that E , running on P^* , extracts all false witnesses with probability q . Then, there exists an expected poly-time machine C , that, given $\text{pp} = (vk, h, \text{setup})$ and access to O_{sig} , where vk, h and setup are a uniformly generated signature verification key, hash function, and initial message for Com respectively, and O_{sig} is an oracle that generates signatures relative to sk chosen when generating vk , outputs one of:*

- $(c, m_1, \tau_1, m_2, \tau_2)$ such that $c = \text{Com}(m_1, \tau_1) = \text{Com}(m_2, \tau_2)$
- (σ, m) such that $\text{Ver}_{vk}(\sigma, m) = 1$ and σ was not received as a signature for m as a response to an oracle query to O_{sig} .
- (m_1, m_2) such that $h(m_1) = h(m_2)$.

with probability $\geq q^3/8 - \text{negl}(n)$, where the probability is over the randomness of C and O .

We can see, given the above lemma, that E can only extract all-false witnesses with negligible probability, else we can easily convert the machine C guaranteed by the lemma into an adversary A that breaks at least one of: the binding property of the commitment scheme, the unforgeability of the signature scheme, or the collision resistance of the CRH family with probability $\geq q^3/24$.

We now continue with the proof of the lemma.

Proof of Lemma 14. We consider the construction of C , which is as follows:

- C takes as inputs $\text{pp} = (vk, h, \text{setup})$ and has oracle access to O_{sig} .

- C simulates a single run of the extractor E on the malicious prover P^* using the externally provided x, pp , and using O_{sig} to create all signatures required in the execution of E . Note, here, that E can straightforwardly be modified to use an externally provided uniformly generated pp , and use a signature oracle instead of generating signatures on its own, and keep the same probability q of extracting all false witnesses (though now this probability is over the choice of the externally generated pp as well).
- As part of E 's execution, C will simulate a single complete interaction between P^* and V , and if P^* succeeds in convincing V in this interaction, then E will, except with negligible probability, extract witnesses from each of the proofs sent by P^* in this interaction. Let W_1 denote the set of witnesses extracted by E .
- If W_1 consists of all-false witnesses, then C rewinds E 's simulated single, complete interaction of (P^*, V) to Stage One, and has E send a different, independently chosen challenge r' as V_1 . It then continues the simulation of E , and, if P^* convinces V in this continuation also, then E , except with negligible probability, also extracts the set of witnesses for this second interaction, which we will refer to as W_2 .
- If W_2 also consists of all-false witnesses, then C uses W_1 and W_2 to either (i) break the binding of the commitment scheme (ii) forge a signature w.r.t vk (iii) find a collision for h .

To perform the last step, C acts as follows:

First recall the types of witnesses that we find in W_1 and W_2 . For the witnesses extracted from Stage Three, it consists of a decommitment to c_3 and

c_0 , showing that c_3 contains sig-com of a tuple $(start, final, M)$, where M is the same TM as that underlying c_0 . For the witness for an oracle proof, it is either a decommitment to c_2 showing that it is of the form $(start, start, M)$, or a decommitment to each of c_1 and c_2 , showing that c_1 contains a sig-com to $(start, current_1, M)$, c_2 contains a tuple $(start, current_2, M)$, together with hash-tree paths showing that $current_2$ is generated from $current_1$ by a single step of M .

First C checks that all commitments present in each witness of W_1 and W_2 is uniquely decommitted over all witnesses in $W_1 \cup W_2$. If C finds some commitment c that is decommitted to two different messages m_1 and m_2 in different witnesses, C outputs c together with the decommitments to m_1 and m_2 and halts.

If C determines that all commitments are uniquely decommitted, then C checks that all signatures present in witnesses of W_1 and W_2 were received as responses to oracle queries on those messages. If C finds that one of the witnesses contains a signature σ and a message m such that σ is a valid signature of m w.r.t vk , but was not received as a response to an oracle query containing m , then C outputs (σ, m) and halts.

If C does not find either of the above, that is, all commitments encountered are uniquely decommitted, and all signatures were honestly received as responses to oracle queries, then C constructs graphs G_1 and G_2 using the witnesses W_1 and W_2 . The nodes of G_1 and G_2 will be labelled with the tuples $(start, current, M)$ that occur in W_1 and W_2 respectively. More specifically, we add a node $(start, current, M)$ if a sig-com to $(start, current, M)$ was received as

a response to a successful oracle query. Notice that for this to happen, the oracle query needed to use a commitment $\text{Com}((start, current, M))$ as its c_2 . We also add a directed edge from node $(start, current_1, M)$ to node $(start, current_2, M)$ if a sig-com of $(start, current_1, M)$ was used in order to obtain a sig-com of $(start, current_2, M)$ from an oracle query using condition (c).

We now note some properties of these graphs (assuming throughout that no commitments were decommitted to two different values, and that all nodes were signed as a response to an oracle query). First, if any node $(start, current_1, M)$ has edges going to two distinct nodes $(start, current_2, M)$ and $(start, current_3, M)$, then that means that P^* managed to convince the oracle that a (deterministic) TM M reaches two different states $current_2$ and $current_3$ starting from $current_1$. Note that the machine M is the same in all three nodes, and in particular, the TM transition function is fixed. Further, the positions of the main tape head and auxiliary tape head before the state transitions are also fixed in the description of $current_1$. Given these fixed values, the only way for P^* both $(start, current_2, M)$ and $(start, current_3, M)$ from $(start, current_1, M)$ is to open up one pair of bits under the main tape head and the auxiliary tape head to show the transition to $(start, current_2, M)$, and a different pair of bits to show the transition to $(start, current_3, M)$. But this means that the witness for $(start, current_2, M)$ must contain a pair of hash tree paths opening up the hashed main tape and auxiliary tape to a particular pair of bits at a particular pair of positions, while the witness for $(start, current_3, M)$ must contain a pair of hash tree paths opening up the same hashed main tape and auxiliary tape, at the same positions, to a *distinct* pair of values. This corresponds to a hash tree collision, which C can retrieve by looking that the hash trees in the witnesses for the two edges. Thus, if there is a

node with out-degree greater than 1, C can find a hash tree collision, which can be converted into a single hash collision.

Also, note that a final state $(start, final, M)$ cannot have any edges leaving it, since this implies the TM continues computing after it should have halted, which means that the witness corresponding to that edge shows that the machine M took a step out of the state $final$, even though the description of M shows that it should halt. Such a witness must be invalid, and thus final states cannot have edges leaving them.

Now C acts as follows: it starts at the node corresponding to the Stage Three witness $(start, final, M)$ in each of G_1 and G_2 . (By our previous assumption that every signature was obtained by an oracle query to O , these witnesses must have been obtained through an earlier oracle query, and since each of the Stage Three witnesses contains a sig-com to some $(start, final, M)$, there will be nodes corresponding to each of these). C then follows the edges from these final nodes backwards, until it reaches a node of the form $(start, start, M)$. To see that we can do this, notice that we can always follow an edge backwards out of a state that isn't the start state, because, by our assumption that no commitments were decommitted in two different ways and that every signature was acquired from an oracle query, the only way to have a node $(start, current, M)$, with $current \neq start$, is if there is some previous node $(start, current', M)$ that was used to sign it. To see that we can't get into cycles, notice that the final state cannot be part of a cycle, since it doesn't have any outgoing edges. If there is a cycle not including the final state, then some node in the cycle must have out-degree > 1 , which,

as discussed earlier, means that C can use that node to find and output a hash collision.

Now suppose C managed to correctly trace backwards in both G_1 and G_2 . That is, we have two paths, going from $(start_1, start_1, M_1)$ to $(start_1, final_1, M_1)$ in G_1 , and from $(start_2, start_2, M_2)$ to $(start_2, final_2, M_2)$ in G_2 . But note that since the Stage Three proof is w.r.t the same c_0 in each of the protocols, since no commitments were decommitted in two different ways, we have that $M_1 = M_2 = M$ where M is the message underlying c_0 , and that $start_1 = start_2 = start$, where $start$ is the starting configuration of M with c_0 on its input tape. But since $r \neq r'$ (except with probability 2^{-n}), we have that $final_1 \neq final_2$. (If $final_1 = final_2$ but $r \neq r'$, then C can simply inspect the Stage Three witnesses, where the entire main tape is revealed, in order to find a hash-tree collision.) That is, we have two paths, each corresponding to *deterministic* computations of M , starting from $start$, and terminating in $final_1$ and $final_2$ respectively. By tracking along these two paths, and inspecting the first node where they diverge, it follows that, at that point, either the bit under the main tape head or under the auxiliary tape head must have been opened up differently in each of the two paths. By examining the corresponding witnesses in W_1 and W_2 , C can extract and output a hash collision.

We now analyze the success probability of C . We consider the transcript of the extraction procedure of E , upto the point it receives message $P_1 = c_0$ from P^* (including any oracle queries made by P^* up to that point). We label as α , the transcript upto the point C receives P_1 , including the choice of

vk, h, setup . We call an α "good" if E extracts all false witnesses with probability at least $q/2$ when continuing the extraction from that α (the probability is over E 's randomness over the remainder of the extraction). Since E extracts all false witnesses with probability q , a good α must be chosen with probability at least $q/2$. With this fact established, we note that with probability $\geq q/2$, we have a good α to start with, and further, that with probability $\geq (q/2)^2$, P^* succeeds in convincing V in both rewind interactions, and C extracts all-false witnesses from each of them as well. (This is because the two transcripts generated by C following the choice of α are distributed in the same way as two independent executions of E conditioned on that α .) Given that both extractions succeeded in obtaining all-false witnesses, C succeeds in breaking one of the commitment scheme, the signature scheme, or the collision resistance of h , except when $r = r'$, which happens with negligible probability. Overall, C succeeds with probability $\geq (q/2)^3 - 2^{-n}$.

□

As noted earlier, the construction of C is sufficient to show that our extractor E cannot extract all-false witnesses except with negligible probability. Hence, from any prover P^* , E must extract a valid witness for $x \in L$ with probability negligibly close to P^* 's success probability.

Thus we have given an extractor for the oracle-aided public-coin protocol given in Fig. 2.9 that runs in expected polynomial time, that, given a prover P^* that convinces V that $x \in L$ with probability p , extracts a witness w for $x \in L$ from P^* with probability $p - \text{negl}(n)$. Hence the protocol is an oracle-aided argument of knowledge.

Returning to the Plain Model

Given the VCO-aided protocol in Fig. 2.9 is a public coin oracle-aided argument of knowledge, we can turn it into a resettably-sound argument of knowledge in the plain model by applying the following steps:

1. We first notice that the table maintained by the stateful oracle in the protocol can be seen as a simulation of a random function F . In particular, we note that given oracle access to a random function, instead of generating a fresh random string every time it encounters a new problem statement and/or first prover message, the oracle could generate its random messages applying a random function F to the problem statement and first prover message, and this alternate implementation would be statistically identical to the one described in the protocol. In particular, if the oracle used a random function, the protocol would still be an argument of knowledge.

We now consider instead an oracle that, rather than using fresh random strings for verifier challenges, instead applies a *PRF* to the problem statement and first prover message. That is, where $O(c_1, c_2, p_1)$ would return a fresh random string, it now returns $r = f(c_1, c_2, p_1)$ for a PRF f . Then the resulting protocol is also an oracle-aided argument of knowledge. To see why, notice that an attacker P^* in the PRF version of the protocol can also be used as an attacker when using the original protocol, with the same success probability (relying on the indistinguishability of the PRF from a truly random function). Thus we can simply use the extractor for the random-function oracle version of this protocol, and apply its guarantee to show that the PRF oracle version is an argument of knowledge.

2. We next apply the PRF transformation of [11] to the public-coin protocol in Fig. 2.9, modified to use the PRF-oracle as described above (recall that in the PRF transformation, we have the verifier generate its random coins in each round by applying a PRF to the current partial transcript). (We note that the [11] transformation also applies to oracle-aided protocols). This makes the protocol in Fig. 2.9 an \mathcal{O} -aided, single instance resettably-sound argument, where \mathcal{O} is the modified PRF oracle.

3. We replace the oracle with a verified-computation slot, to get the protocol in Fig. 2.4. Since the honest prover never uses the oracle, completeness still holds in this modified protocol. By a proof analogous to the one described in Section 2.5, this transformed protocol is also a single-instance resettably-sound argument of knowledge in the plain model, since an attacker for the protocol in the plain model can be transformed into an attacker for the \mathcal{O} -aided version of the protocol, for \mathcal{O} as the modified PRF oracle. Roughly, a resetting attacker P^* for the single-instance oracle-free version of the protocol in Fig. 2.4 can be turned into an attacker \tilde{P}^* for the oracle-aided version of the protocol in Fig. 2.9, where \tilde{P}^* forwards all messages that would be sent to the Stage Two verified computation slot to the VCO instead, and forwards the oracle challenges back to P^* . Note that the resetting attacker may send multiple concurrent Stage Two messages in an interleaved fashion, but since our oracle accepts concurrent proofs, the attacker's messages can be forwarded without issues. In fact, this forwarding strategy results in an exact simulation of P^* , and so \tilde{P}^* succeeds with the same probability as P^* . Thus we can extract from P^* by constructing \tilde{P}^* and applying the extractor for the oracle-aided version of the protocol to it. This shows that the protocol in Fig. 2.4 is single-instance

resetably sound.

4. By applying Claim 2, the single-instance resetably-sound argument of knowledge can be turned into a full-fledged resetably-sound argument of knowledge.

Taken all together, we complete the proof that the protocol in Fig. 2.4 is a constant-round, full-fledged resetably-sound AOK.

2.6.2 Proof of Zero Knowledge property

In the case of zero knowledge, we follow a proof strategy similar to the protocol discussed in Section 2.4.2. That is, we first assume that the simulator has access to a weak VC oracle, that is just sufficient for it to generate a false witness for Stage Three. More precisely, we define *Valid Verified Computation Oracles* \mathcal{O} below, and show that the simulator succeeds given access to such an oracle, and further, that the simulator can create a valid verified computation oracle by rewinding any malicious verifier that completes the protocol with non-negligible probability.

As in the definition of VCOs, our oracle is going to take as input a pair (c_1, c_2) , and return a signature on c_2 if c_2 is a commitment to a starting configuration, or else a configuration generated by one step of computation from the configuration underlying c_1 . However, unlike the previous definition of VCOs, where the oracle was provided a WI proof that c_2 satisfies one of these conditions, we instead require that our weak *Valid VCOs* be supplied with an *explicit* witness for one of the two conditions.

To see why we need an explicit witness, we recall the proof structure in Section 2.4.2. There, we showed first that the protocol can be straight-line simulated if we just have access to a *valid* oracle, and second showed that a simulator can simulate a valid oracle by rewinding V^* 's signature slot. We will do something similar here, showing that given a *valid* VCO, the protocol can be straight-line simulated, and then showing that the simulator can rewind any malicious verifier's *verified computation slot* to simulate a *valid* VCO. However, when simulating a valid VCO using the malicious verifier's verified computation slot, the simulator may have to rewind the malicious verifier many times, sending a fresh WI proof in each rewinding, showing that (c_1, c_2) satisfies one of the two conditions (c_2 is a starting configuration, or is generated from c_1 by one step of computation), repeating with re-randomized versions of this proof until the verifier accepts and returns a signature. The simulator cannot, in general, generate these re-randomized proofs without some extra information such as a witness. So, in order to allow the simulator generate these multiple WI-proofs, we require that every oracle query must contain an explicit witness w for one of the two conditions. (Alternatively, we could have required that the proofs themselves be re-randomizable without a witness, but we do not consider this alternative here.)

We now give a formal definition of *Valid VCOs*

Definition 23 (Valid Verified Computation Oracle). *Let SIG be a strong signature scheme, Com be a commitment scheme, and $\{\mathcal{HF}_n\}_{n \in \mathcal{N}}$ a collision-resistant hash function family.*

We say that O is a Valid Verified Computation Oracle w.r.t $\text{pp} = (\text{vk}, h, \text{setup})$, if, whenever O is given a pair of commitments (c_1, c_2) as input, together with an explicit

witness w , such that:

- a) w contains a decommitment of each of c_1 and c_2 , also showing that $c_2 = \text{Com}((\text{start}, \text{start}, M), \tau_2)$, where start is a starting TM configuration, M is a TM description, and τ_2 is commitment randomness, or
- b) w is such that $(\langle c_1, c_2, \text{vk}, h \rangle, w) \in \mathbf{R}_{L_3}$ (See Fig 2.6),

then, with overwhelming probability O returns (σ, c'_2, τ'_2) , where $\text{Ver}_{\text{vk}}(\sigma, c'_2) = 1$, and if $c_2 = \text{Com}(m_2, \tau_2)$ for some m_2 and τ_2 , then $c'_2 = \text{Com}(m_2, \tau'_2)$.

We point out again that the oracle here takes an explicit witness w as input (as opposed to a WI-AOK with respect to that witness). Secondly, we also emphasize that the signature returned on c_2 could in fact be on a different commitment c'_2 for the message m underlying c_2 . This is again because our simulator has to simulate a valid VCO by rewinding an uncooperative malicious V^* , and may need to try several different commitments for m before it can successfully receive a signature from V^* . Note that we explicitly provide a decommitment to c_1 and c_2 in each of the witnesses, and thus the simulator can, in fact, generate different commitments to m .

We now show that, given such a weak Verified Computation Oracle, a straight-line simulator S can complete the proof in Stage Three, as follows:

- S generates a TM description $M = V^*$ corresponding to the verifiers code and auxiliary input, and committing to M as c_0 in Stage One.
- S generates a TM configuration start corresponding to a starting configuration of M , with c_0 as the input tape, and using the valid Verified Computation Oracle to get a sig-com for $(\text{start}, \text{start}, M)$ using option (a).

- S uses option (b) of the oracle to generate sig-coms for $(start, current, M)$, for successive states $current$ in the computation of M , until a final state is reached. Note that, for this step, the simulator will have to repeatedly and efficiently produce a description of a configuration, say $current_2$, derived from a configuration say $current_1$, by running a single step of M . Recall that in order to do so, it must efficiently produce a hash tree of a tape for $current_2$, appropriately updated from running a single step of M from $current_1$.

However, since the two tapes only differ in a single bit, this update can indeed be done efficiently (in time proportional to the depth of the tree) by simply changing the required bit in $current_1$'s hash tree, and updating its parent nodes up to the root.

- S uses the sig-com on the final state, $(start, final, M)$, as a trapdoor witness for Stage Three of the proof.

We also observe that indistinguishability of the transcripts produced by the simulator holds by a standard hybrid argument, using the following hybrids:

H_0 The hybrid corresponding to the transcripts output by the simulator, where

- The commitment c_0 contains the code of the verifier V^* .
- The commitment c_2 contains a message of the form $(start, start, V^*)$
- The proof in Stage Two uses a witness for condition (b)
- The commitment c_3 in Stage 3 contains a message of the form $(start, final, V^*)$.

– The proof in Stage Three uses a witness for condition 2.

H_1 Identical to H_0 , except the proof in Stage Three uses a witness w for $x \in L$ to complete the proof using condition 1.

H_2 Identical to H_1 , except the commitment c_3 is replaced with a commitment to 0^{6n} .

H_3 Identical to H_2 , except the proof in Stage Two uses a witness w for $x \in L$ to complete the proof using condition (a).

H_4 Identical to H_2 , except the commitment c_2 is replaced with a commitment to 0^{6n} .

H_5 The hybrid corresponding to the transcripts generated by an interaction with the honest prover. Identical to H_4 , except the commitment c_0 is replaced with a commitment to 0^{6n} .

Note that the pairs of hybrids (H_0, H_1) and (H_2, H_3) are indistinguishable based on the witness indistinguishability of the proofs used in Stages Two and Three, and that the pairs of hybrids (H_1, H_2) , (H_3, H_4) and (H_4, H_5) are indistinguishable based on the hiding property of Com. Thus we have that H_0 is indistinguishable from H_5 , showing that the transcript generated by a simulator is indistinguishable from a transcript from an interaction with an honest prover.

To show zero knowledge in the plain model (without this weak signing oracle), just as in Lemma 11 (see also Section 2.5), we show that we can almost exactly simulate a weak oracle in the plain model, by using Stage Two of the protocol. Given a malicious verifier V^* , the simulator first honestly simulates a single instance of the protocol in Fig. 2.4 for V^* up to the end of Stage 2, using

option (b) to complete the proof in Stage 2 using any arbitrarily chosen TM M and any arbitrary starting state in m_2 . If this instance would cause the prover to abort, the simulator outputs the transcript and also aborts. Otherwise, it follows the strategy in Lemma 11 to create a valid VC oracle, and uses it to continue and complete Stage 3 of the protocol using option 2 of the WI-AOK.

Recapping the strategy from Lemma 11, the simulator rewinds V^* to the start of Stage Two, and repeatedly sends new proofs for option (b) of the WI-AOK, using fresh commitments to arbitrary machines M and starting states s . It does so until it receives $2n$ valid signatures from V^* , and lets t be the number of rewindings taken. Then, it simulates the valid VC oracle as follows: on oracle query (c_1, c_2, w) , the simulator rewinds V^* to the start of Stage 2, and generates fresh commitments to the messages underlying c_1 and c_2 . (In order to generate different commitments of c_1 and c_2 , the simulator needs to know the messages underlying these commitments. But note that the caller of the oracle provides an explicit witness as part of its oracle queries, containing decommitments to each of c_1 and c_2 .) The simulator uses the witness w to send a WI proof for Stage Two to V^* . It repeats this step up to t times until V^* completes Stage Two and signs one of the commitments c'_2 sent by the simulator. In this case, the simulator responds to the oracle query with the signature σ , the commitment c'_2 , as well as the decommitment randomness τ'_2 for c'_2 . If V^* fails in all t rewindings, the simulator returns \perp for the oracle query.

Following the analysis Lemma 11, it can be seen that for any V^* that completes the first instance of Stage Two with non-negligible probability, this simulator correctly simulates a valid VCO, and runs in expected polynomial time, with negligible probability of returning \perp . To complete the analysis, we use the

fact that the commitments at the start of Stage 2 must be hiding, and that the proofs in Stage Two must be witness indistinguishable. This will guarantee that a malicious verifier will complete Stage Two with the same probability p (up to a negligible factor), no matter if the messages underlying the commitments c_1, c_2 are changed, or if the option (a), (b) or (c) used to complete the WI-AOK in Stage Two is altered.

Combining the arguments above, we have that the protocol in Fig 2.4 can be simulated with access to a valid VC oracle, and further, that valid VC oracle can be simulated for any malicious verifier that completes Stage Two with non-negligible probability, while all other malicious verifiers can be trivially simulated. Thus the protocol has a simulator in the plain model and is zero knowledge. Combined with the preceding subsections, we conclude that it is a constant-round, full-fledged resettably sound argument of knowledge, completing the proof in the section.

2.7 Applications

We now discuss some applications of our construction of resettably-sound zero-knowledge arguments of knowledge. By plugging our protocol into the constructions of [41, 111, 11], with some minor modifications that we discuss shortly, we immediately obtain the following theorem. Roughly speaking, in a resettably-witness indistinguishable (resp., zero-knowledge) argument, the witness-indistinguishability (resp., zero-knowledge) property is required to hold also in the presence of a resetting verifier. Resettably zero-knowledge is a stronger property than concurrent zero-knowledge, since not only can a re-

setting verifier interact with multiple provers, but can also force any prover to restart its interaction with the same random tape. See [41, 11] for formal definitions.

Theorem 15. *Assume the existence of one-way functions. Then,*

- *there exists a constant-round resettably-witness-indistinguishable argument of knowledge for all of NP,*
- *there exists a $\tilde{O}(\log n)$ -round resettably-zero-knowledge argument of knowledge for all of NP.*

The construction of a constant-round resettably-witness-indistinguishable argument of knowledge follows directly from the results of [11]. For the construction of a resettably-zero-knowledge argument of knowledge, recall that [11] (following [41]) construct resettably-zero-knowledge arguments of knowledge by compiling (using a resettably-sound zero-knowledge argument of knowledge) some underlying concurrent zero-knowledge protocols of the “committed-verifier type” where the verifier commits to its “challenges” at the beginning of the protocol, and then reveals them one by one in sequential “slots”. The underlying concurrent zero-knowledge protocol, however, needs the commitment to be statistically-hiding, in order to make the protocol a zero knowledge *proof* (as opposed to an argument). We first note that we can also use computationally-hiding commitments (that exists based on one-way functions) in the zero-knowledge protocol of [111], at the cost of making the protocol a rZK argument, not a proof as in [111]. Further, by a minor tweak, we can also make the protocol an argument of knowledge. More precisely, we change it so that the verifier in the final stage of the protocol, instead of opening up all

the computationally hiding commitments, simply reveals the committed values and provides a *resettable-sound* zero-knowledge argument of knowledge of the value. The resettable-soundness is needed for security against the resetting verifier, who is sending the proof in this step, and the argument of knowledge property is needed to provide a concrete reduction to the security of the commitment scheme, so that if the verifier breaks the binding of its commitments, we can explicitly extract two different openings for the commitment.

The overall protocol is now an argument of knowledge, since the extractor can now simulate the proof sent by the verifier in the last stage, to convincingly open the commitment to two different values, allowing us to extract the witness from the prover.

We point out that the rsZK-AOK is used here in a black-box way, and, in particular, we do not need the verifier to commit to the coins it uses for the rsZK-AOK in the initial commitment phase of the overall protocol.

We also note that the trick of replacing statistically-hiding commitments with computationally-hiding commitments, but using just a “plain”, as opposed to resettable-sound, zero-knowledge argument of knowledge in the final stage in the protocol, was used in [111] to get a concurrent zero-knowledge argument from one-way functions. The soundness and proof-of-knowledge property of our protocol follows in exactly the same way as the corresponding protocol in [111], a full proof for which appears in [114]. We note that since we use a *resettable-sound* AOK in the last step, we additionally preserve the resettable-zero-knowledge property, which is lost in [111] because their “plain” ZK argument in the final stage is not secure against resetting verifiers.

Acknowledgements

We are very grateful to Ran Canetti for pointing out the connection to [42].

CHAPTER 3
INDISTINGUISHABILITY OBFUSCATION FROM
SEMANTICALLY-SECURE MULTILINEAR ENCODINGS

This chapter contains joint work with Rafael Pass (Cornell University) and Sidharth Telang (Cornell University). It appeared in the proceedings of the International Cryptology Conference (CRYPTO) 2014.

3.1 Introduction

The goal of *program obfuscation* is to “scramble” a computer program, hiding its implementation details (making it hard to “reverse-engineer”), while preserving the functionality (i.e, input/output behavior) of the program. Precisely defining what it means to “scramble” a program is non-trivial: on the one hand, we want a definition that can be plausibly satisfied, on the other hand, we want a definition that is useful for applications.

A first formal definition of such program obfuscation was provided by Hada [79]: roughly speaking, Hada’s definition—let us refer to it as *strongly virtual black-box*—is formalized using the simulation paradigm. It requires that anything an attacker can learn from the obfuscated code, could be simulated using just black-box access to the functionality.¹ Unfortunately, as noted by Hada, only learnable functionalities can satisfy such a strong notion of obfuscation: if the attacker simply outputs the code it is given, the simulator must be able to recover the code by simply querying the functionality and thus the functionality must be learnable.

¹Hada actually considered a slight distributional weakening of this definition.

An in-depth study of program obfuscation was initiated in the seminal work of Barak, Goldreich, Impagliazzo, Rudich, Sahai, Vadhan, and Yang [12]. Their central result shows that even if we consider a more relaxed simulation-based definition of program obfuscation—called *virtual black-box (VBB)* obfuscation—where the attacker is restricted to simply outputting a single bit, impossibility can still be established.² Their result is even stronger, demonstrating the existence of families of functions such that given black-box access to f_s (for a randomly chosen s), not even a *single* bit of s can be guessed with probability significantly better than $1/2$, but given the code of any program that computes f_s , the entire secret s can be recovered. Thus, even quite weak simulation-based notions of obfuscation are impossible.

But weaker notions of obfuscation may be achievable, and may still suffice for (some) applications. Indeed, Barak *et al.* [12] also suggested two such notions:

- The notion of *indistinguishability obfuscation*, first defined by Barak *et al.* [12] and explored by Garg, Gentry, Halevi, Raykova, Sahai, and Waters [57], roughly speaking requires that obfuscations $O(C_1)$ and $O(C_2)$ of any two *equivalent* circuits C_1 and C_2 (i.e., whose outputs agree on all inputs) from some class C are computationally indistinguishable.
- The notion of *differing-input obfuscation*, first defined by Barak *et al.* [12] and explored by Boyle, Chung and Pass [32] and by Ananth, Boneh, Garg, Sahai and Zhandry [1] strengthens the notion of indistinguishability obfuscation to also require that even if C_1 and C_2 are not equivalent circuits, if an attacker can distinguish obfuscations $O(C_1)$ and $O(C_2)$, then the at-

²A similar notion of security (without referring to obfuscation) was considered even earlier by Canetti [40] in the special case of what is now referred to as *point-function obfuscation*.

tacker must “know” an input x such that $C_1(x) \neq C_2(x)$, and this input can be efficiently “extracted” from the attacker.

In a very recent breakthrough result, Garg, Gentry, Halevi, Raykova, Sahai, and Waters [57] provided the first candidate constructions of indistinguishability obfuscators for all polynomial-size circuits, based on so-called *multilinear (a.k.a. graded) encodings* [28, 115, 54]—for which candidate constructions were recently discovered in the seminal work of Garg, Gentry and Halevi [54], and more recently, alternative constructions were provided by Coron, Lepoint and Tibouchi [49].

The obfuscator construction of Garg et al proceeds in two steps. They first provide a candidate construction of an indistinguishability obfuscator for NC^1 (this construction is essentially assumed to be secure); next, they demonstrate a “bootstrapping” theorem showing how to use fully homomorphic encryption (FHE) schemes [60] and indistinguishability obfuscators for NC^1 to obtain indistinguishability obfuscators for all polynomial-size circuits. Further constructions of obfuscators for NC^1 were subsequently provided by Brakerski and Rothblum [36] and Barak, Garg, Kalai, Paneth and Sahai [9]—in fact, these constructions achieve the even stronger notion of virtual-black-box obfuscation in idealized “generic” multilinear encoding models. Additionally, Boyle, Chung and Pass [32] present an alternative bootstrapping theorem, showing how to employ differing-input obfuscations for NC^1 to obtain differing-input (and thus also indistinguishability) obfuscation for both circuits and Turing machines. (Ananth et al [1] also provide Turing machine differing-input obfuscators, but start instead from differing-input obfuscators for polynomial-size circuits.)

In parallel with the development of candidate obfuscation constructions,

several surprising applications of both indistinguishability and differing-input obfuscations have emerged (see e.g., [57, 116, 81, 31, 56, 32], and more recently [18, 33, 71, 89, 88]). Most notable among these is the work of Sahai and Waters [116] (and the “punctured program” paradigm it introduces) which shows that for some interesting applications of virtual black-box obfuscation (such as turning private-key primitives into public-key one), the weaker notion of indistinguishability obfuscation suffices. Furthermore, as shown by Goldwasser and Rothblum [77], indistinguishability obfuscators provide a very nice “best-possible” obfuscation guarantee: if a functionality can be VBB obfuscated (even non-efficiently!), then any indistinguishability obfuscator for this functionality is VBB secure. Finally, as shown by Boyle, Chung and Pass [32] indistinguishability obfuscation in fact implies a notion of differing-input obfuscation (when restricted to programs that differ on polynomially-many inputs); and this notion already suffices for some applications of differing-input obfuscation (see e.g., [15], [38], [39]).

3.1.1 Towards “Provably-Secure” Obfuscation

But despite these amazing developments, the following question remains open:

Can the security of general-purpose indistinguishability obfuscator be reduced to some “natural” intractability assumption?

The principal goal of the current paper is to make progress toward addressing this question. Note that while the construction of indistinguishability obfuscation of Garg et al is based on *some* intractability assumption, the assumption

is very tightly tied to their scheme—in essence, the assumption stipulates that their scheme is a secure indistinguishability obfuscator. The VBB constructions of Brakerski and Rothblum [36] and Barak et al [9] give us more confidence in the plausible security of their obfuscators, in that they show that at least “generic” attacks—that treat multilinear encoding as if they were “physical envelopes” on which multilinear operations can be performed—cannot be used to break security of the obfuscators. But at the same time, non-generic attacks against their scheme are known—since general-purpose VBB obfuscation is impossible. Thus, it is not clear to what extent security arguments in the generic multilinear encoding model should make us more confident that these constructions satisfy e.g., a notion of indistinguishability obfuscation. In particular, the question of to what extent one can capture “real-world” security properties from security proofs in the generic model through a “meta-assumption” (regarding multilinear encodings) was raised (but not investigated) in [9]; see Remark 1 there.

In this work, we initiate a study of the above-mentioned question:

- We are concerned with the question of whether some *succinct* and *general* assumption (that is interesting in its own right, and is not “tailored” to a particular obfuscation construction) about some *low-level primitive* for which candidate constructions are known (e.g., multilinear encodings), can be used to obtain indistinguishability obfuscation.
- More importantly, we are interested in *reducing* the security of the obfuscation to some *simpler* assumption, not just in terms of “description size” but in terms of computational complexity—that is, we are not interested in assumptions that “directly” (without any security reduction) imply security

of the obfuscation.

- Finally, ideally, we would like the assumption to be *efficiently falsifiable* [99], so that it is possible to efficiently check whether the assumption is broken. This is particularly pressing since the assumption that a particular scheme (e.g., one of the schemes of [57, 36, 9]) is an indistinguishability obfuscator is not an efficiently falsifiable assumption, making it hard to check whether they can be broken or not: a presumed attacker must exhibit two *functionally-equivalent* circuits C_1 and C_2 that it can distinguish obfuscations of; but checking whether two circuits are functionally equivalent may not be polynomial-time computable. (In fact, assuming the existence of indistinguishability obfuscation and one-way functions, it is easy to come up with a method to sample C_1, C_2, z such that with high probability $C_1(z) \neq C_2(z)$ (and thus, given z , we can easily distinguish obfuscations of them), yet the pair of circuits (C_1, C_2) are indistinguishable from a pair of functionally equivalent circuits.³ Thus, there are “fake attacks” on indistinguishability obfuscation that cannot be efficiently distinguished from a real attack.)

3.1.2 Security of Multilinear (Graded) Encodings

Towards explaining the assumptions we consider, let us start by briefly recalling multilinear (a.k.a. graded) encoding schemes [54, 57]. Roughly speaking, such

³In particular, (mirroring the ideas from the lower bound for witness encryption of [59]), given a statement x , let C_b^x be an obfuscation of a circuit that given a witness w outputs b iff w is an NP-witness for the statement x (and \perp otherwise). If x is false, then by the indistinguishability obfuscation property, (C_0^x, C_1^x) is indistinguishable from two obfuscations of the *same* constant \perp function. This still holds even if we sample a true x (and its associated witness z) from a hard-on-the-average language (as long as we do not give z to the distinguisher). Yet given the trapdoor z , we can clearly distinguish C_0^x, C_1^x and also obfuscations of them.

schemes enable anyone that has access to a *public parameter* pp and *encodings* $E_S^x = \text{Enc}(x, S)$, $E_{S'}^y = \text{Enc}(y, S')$ of ring elements x, y under the sets $S, S' \subset [k]$ to *efficiently*:⁴

- compute an encoding $E_{S \cup S'}^{x \cdot y}$ of $x \cdot y$ under the set $S \cup S'$, as long as $S \cap S' = \emptyset$;
- compute an encoding E_S^{x+y} of $x + y$ under the set S as long as $S = S'$;
- compute an encoding E_S^{x-y} of $x - y$ under the set S as long as $S = S'$.

(Given just access to the public-parameter pp , generating an encoding to a particular element x may not be efficient; however, it can be efficiently done given access to the *secret parameter* sp .) Additionally, given an encoding E_S^x where the set S is the whole universe $[k]$ —called the “target set”—we can efficiently check whether $x = 0$ (i.e., we can “zero-test” encodings under the target set $[k]$.) In essence, multilinear encodings enable computations of certain restricted set of arithmetic circuits (determined by the sets S under which the elements are encoded) and finally determine whether the output of the circuit is 0; we refer to these as the *legal arithmetic circuits*.

Semantical Security of Multilinear (Graded) Encodings The above description only explains the *functionality* of multilinear encodings, but does not discuss *security*. As far as we are aware, there have been two approaches to defining security of multilinear encodings. The first approach, initiated in [54], stipulates specific hardness assumptions closely related to the DDH assumption. The second approach instead focuses on *generic attackers* and assumes that the attacker does

⁴Just as [36, 9], we here rely on “set-based” graded encoding; these were originally called “generalized” graded encodings in [54]. Following [57, 9] (and in particular the notion of a “multilinear jigsaw puzzles” in [57]), we additionally enable anyone with the secret parameter to encode *any* elements (as opposed to just *random* elements as in [54]).

not get to see the actual encodings but instead can only access them through legal arithmetic circuits.

In this work, we consider the first approach, but attempt to capture a general *class* of algebraic “decisional” assumptions (such as the the graded DDH assumption of [54]) which hold against generic attackers (and as such, it can be viewed as a merge of the two approaches). In essence, our notion of (single-message) *semantical security* attempts to capture the intuition that encodings of elements m_0 and m_1 (under the set S) are indistinguishable in the presence of encodings of “auxiliary” elements \vec{z} (under sets \vec{T}), as long as m_0, m_1, \vec{z} are sampled from *any* “nice” distribution D ; in the context of a graded DDH assumption, think of \vec{z} as a vector of independent uniform elements, m_0 as the product of the elements in \vec{z} and m_1 as an independent uniform element. We analogously consider stronger notions of *constant-message* and *multi-message* semantical security, where m_0, m_1 (and S) are replaced by either constant-length or arbitrary polynomial-length vectors \vec{m}_0, \vec{m}_1 of elements (and sets \vec{S}).

Defining what makes a distribution D “nice” turns out to be quite non-trivial: A first (and minimal) approach—similar to e.g., the uber assumption of [25] in the context of bilinear maps—would be to simply require that D samples elements $\vec{m}_0, \vec{m}_1, \vec{z}$ such that no generic attacker can distinguish \vec{m}_0, \vec{z} and \vec{m}_1, \vec{z} . As we discuss in Section 3.1.3, the most natural formalization of this approach can be attacked assuming standard cryptographic hardness assumptions. The distribution D considered in the attack, however, is “unnatural” in the sense that encodings of \vec{m}_b, \vec{z} actually leak information about \vec{m}_b even to generic attackers (in fact, this information fully determines the bit b , it is just that it cannot be computed in polynomial time).

Our notion of a *valid* message distribution disallows such information leakage w.r.t. generic attacks. More precisely, we require that every (even unbounded-size) legal arithmetic circuit C is *constant* over (m_b, \vec{z}) , $b \in \{0, 1\}$ with overwhelming probability; that is, there exists some bit c such that with overwhelming probability over $m_0, m_1, \vec{z} \leftarrow D$, $C(m_b, \vec{z}) = c$ for $b \in \{0, 1\}$ (recall that a legal arithmetic circuit needs to end with a zero-test and thus the output of the circuit will be either 0 or 1). We refer to any distribution D satisfying this property as being *valid*, and our formal definition of semantical security now only quantifies over such valid message distributions.

Obfuscation from Semantically-Secure Multilinear Encodings As a starting point, we observe that slight variants of the constructions of [36, 9] can be shown to satisfy indistinguishability obfuscation for NC¹ assuming *multi-message* semantically-secure multilinear encodings. In fact, any VBB secure obfuscation in the generic model where the construction only releases encodings of elements (as the constructions of [36, 9] do) satisfies indistinguishability obfuscation assuming a slight strengthening of multi-message semantical security where validity only considers *polynomial-size* (as opposed to arbitrary-size) legal arithmetic circuits:⁵ let \vec{m}_0 denote the elements corresponding to an obfuscation of some program Π_0 , and \vec{m}_1 the elements corresponding to an obfuscation of some functionally equivalent program Π_1 . VBB security implies that all polynomial-size legal arithmetic circuits are constant with overwhelming probability over both \vec{m}_0 and \vec{m}_1 (as any such query can be simulated given black-box access to the functionality of the program), and thus encodings of \vec{m}_0 and \vec{m}_1 (i.e., obfuscations of Π_0 and Π_1) are indistinguishable. By slightly tweak-

⁵We thank Sanjam Garg for this observation.

ing the construction of [9] and the analysis⁶, we can extend this to hold against *all* (arbitrary-size) legal arithmetic circuits, and thus indistinguishability of the encodings (which implies indistinguishability of the obfuscations) follows as a direct consequence of the multi-message security assumption.

While this observation does takes us a step closer towards basing the security of obfuscation on a simple, natural, assumption, it is unappealing in that the assumption itself directly implies the security of the scheme (without any security reduction); that is, it does not deal with our second desiderata of *reducing* security to a *simpler* assumption—in particular, simply assuming that the (slight variant of the) scheme of [9] is secure is a special case of the multi-message security assumption.

Our central result shows how to construct indistinguishability obfuscators for NC^1 based on the existence of *constant-message* semantically-secure multilinear encodings; in the sequel, we simply refer to such schemes as being semantically secure (dropping “constant-message” from the notation). Note that the constant-message restriction not only simplifies (and reduces the complexity) of the assumption, it also takes us a step closer to the more standard GDDH assumption. (As far as we know, essentially all DDH-type assumptions in “standard”/bilinear or multilinear settings consider a constant-message setting, stipulating indistinguishability of either a *single* or a *constant* number of elements in the presence of polynomially many auxiliary elements. It is thus safe to say that such constant-message indistinguishability assumptions are significantly better understood their multi-message counterpart.)

Theorem 16 (Informally stated). *Assume the existence of semantically secure multi-*

⁶Briefly, we need to tweak the construction to ensure a “perfect” simulation property.

linear encodings. Then there exists an indistinguishability obfuscator for NC^1 .

As far as we know, this is the first result presenting indistinguishability obfuscators for NC^1 based on any type of assumption with a “non-trivial” security reduction w.r.t. arbitrary nuPPT attackers.

The core of our result is a general technique for transforming any obfuscator for matrix branching programs that satisfies a weak notion of *neighboring-matrix* indistinguishability obfuscation—which roughly speaking only requires indistinguishability of obfuscations of branching programs that differ only in a constant number of matrices—into a “full-fledged” indistinguishability obfuscator. (We emphasize that this first result is unconditional—it does not pertain to any particular construction and does not rely on any computational assumptions—and we thus hope it may be interesting in its own right.) We next show how to adapt the construction of [9] and its analysis to satisfy neighboring-matrix indistinguishability obfuscation based on semantically secure multilinear encodings; on a high-level, the security analysis in the generic model is useful for proving that the particular message distribution we consider is “valid”.⁷

If additionally assuming the existence of a leveled FHE [112, 60] with decryption in NC^1 —implied, for instance, by the LWE assumption [37, 34]—this construction can be bootstrapped up to obtain indistinguishability obfuscators for all polynomial-size circuits by relying on the technique from [57].

Theorem 17 (Informally stated). *Assume the existence of semantically secure multilinear encodings and a leveled FHE with decryption in NC^1 . Then there exists indistin-*

⁷As we explain in more details later, to use our transformation, we need to deal with branching programs that satisfy a slightly more liberal definition of a branching program than what is used in earlier works. This is key reason why we need to modify the construction and analysis from [9].

guishability obfuscators for P/poly.

Semantical Security w.r.t. Restricted Classes of Distributions Our most basic notion of semantical security requires indistinguishability to hold w.r.t. to *any* “valid” message distribution. This may seem like a strong assumption. Firstly, such a notion can clearly not be satisfied by a *deterministic* encoding schemes (as envisioned in the original work of [28])—we can never expect encodings of 0 and 1 (under a non target set, and without any auxiliary inputs) to be indistinguishable. Secondly, even if we have a randomized encoding scheme in mind (such as the candidates of [54, 49]), giving the attacker access to encodings of *arbitrary* elements may be dangerous: As mentioned in [54], attacks (referred to as “weak discrete logarithm attacks”) on their scheme are known in settings where the attacker can get access to “non-trivial” encodings of 0 under any *non-target* set $S \subset [k]$. (We mention that, as far as we know, no such attacks are *currently* known on the candidate construction of [49].)

For the purposes of the results in our paper, however, it suffices to consider a notion of semantical security w.r.t. *restricted classes of distributions* D . In particular, to deal with both of the above issues, we consider “high-entropy” distributions D that sample elements $\vec{m}_0, \vec{m}_1, \vec{z}$ such that 1) each individual element has high-entropy, and 2) any element, associated with a *non-target* set $S \subset [k]$, that can be obtained by applying “legal” algebraic operations to (\vec{m}_b, \vec{z}) (for $b \in \{0, 1\}$) has high-entropy (and thus is non-zero with overwhelming probability).⁸ We refer to such message distributions as being *entropically valid*.

Basing Security on a Single Falsifiable Assumption The assumption that a

⁸Technically, by high-entropy, we here mean that the min-entropy is at least $\log |R| - O(\log \log |R|)$ where R is the ring associated with the encodings; that is, the min-entropy is “almost” optimal (i.e., $\log |R|$).

scheme satisfies semantical security may be viewed as an (exponential-size) *class* of algebraic “decisional” assumptions (or as a “meta”-assumption, just like the “uber assumption” of [25]): we have one assumption for each valid message distributions D . Indeed, to prove indistinguishability of obfuscations of two circuits C_0, C_1 , we rely on an instance in this class that is a function of the circuits C_0, C_1 —in the language of [59, 65], security is thus based on an “instance-dependent” assumption.

This view-point also clarifies that semantical security is not an *efficiently falsifiable* assumption [99]: the problem is that there may not exist an efficient way of checking whether a distribution D is valid (as this requires checking that *all* legal arithmetic circuits are constant with overwhelming probability, which in our particular case would require checking whether C_0 and C_1 are functionally equivalent).

We finally observe that both of these issues can be overcome if we make subexponential hardness assumptions: there exists a single (uniform PPT samplable) distribution Sam over nuPPT message distributions D that are *provably* entropically valid such that it suffices to assume the existence of an encoding scheme that is entropic semantically secure w.r.t., this particular distribution with *subexponentially small indistinguishability gap*.⁹ Note that this is a single, non-interactive and efficiently falsifiable, decisional assumption.

⁹These results were added to our e-Print report April 25, 2014, motivated in part by [65] (which bases witness encryption [59] on an instant-independent assumption) and a question asked by Amit Sahai.

3.1.3 Alternative Security Notions for Multilinear Encodings

We finally investigate various ways of defining a “super” (or uber) assumption for multilinear encodings. As mentioned above, a natural way of defining security of multilinear encodings would be to require that for specific classes of problems, generic attacks cannot be beaten (this is the approach alluded to in [9]). Perhaps the most natural instantiation of this in the context of a multilinear DDH assumption would be to require that for any distribution D over $\vec{m}_0, \vec{m}_1, \vec{z}$ (where \vec{m}_0, \vec{m}_1 are constant-length sequences), if encodings of \vec{m}_0, \vec{z} and \vec{m}_1, \vec{z} are indistinguishable w.r.t. to generic attackers, then they are also indistinguishable w.r.t. arbitrary nuPPT attackers; in essence, “if an algebraic decisional assumption holds w.r.t. to generic attacks, then it also holds with respect to nuPPT attackers”. We refer to this notion of security as *extractable uber security*.¹⁰

Our second main result shows that, assuming the existence of a leveled FHE with decryption in NC^1 , there do not exist extractable uber-secure multilinear encodings (even if we only require security to hold w.r.t high-entropy distributions D).

Theorem 18 (Informally stated). *Assume the existence of a leveled FHE with decryption in NC^1 . Then no multilinear encodings can satisfy extractable (entropic) uber security.*

The high-level idea behind this result is to rely on the “conflict” between the feasibility of VBB obfuscation in the generic model of [9] and the impossibility of VBB obfuscation in the “standard” model [12]: we let \vec{m}_b, \vec{z} contain a generically-

¹⁰We use the adjective “extractable” as this security notion implies that if an nuPPT attacker can distinguish encodings, then the arithmetic circuits needed to distinguish the elements can be efficiently extracted out.

secure VBB obfuscation of a program Π_b that hides b given just black-box access to Π_b , yet b can be recovered given the code of Π_b . By generic security of the obfuscation, it follows that *efficient* generic attackers cannot distinguish \vec{m}_0, \vec{z} and \vec{m}_1, \vec{z} yet, “non-generic” (i.e., standard PPT) attackers can. In our formal treatment, to rule out *constant-message* (as opposed to multi-message) security, we rely on a variant of the obfuscator presented in this paper, enhanced using techniques from [9].

We emphasize that in the above attack it is crucial that we restrict to efficient (nuPPT) generic attacks. We finally consider several plausible ways of defining uber security for multilinear encodings, which circumvent the above impossibility results by requiring indistinguishability of encodings only if the encodings are *statistically* close w.r.t. *unbounded* generic attackers (that are restricted to making polynomially many zero-test queries). We highlight that none of these assumptions are needed for our construction of an indistinguishability obfuscation and are stronger than semantical security, but they may find other applications.

3.1.4 Construction Overview

The Basic Obfuscator We start by providing a construction of a “basic” obfuscator; our final construction will then rely on the basic obfuscator as a black-box. The construction of this obfuscator closely follows the design principles laid out in the original work by Garg et al [57] and follow-up constructions [36, 9] (in fact, the basic obfuscator may be viewed as a simplified version of the obfuscator from [9]). As these works, we proceed in three steps:

Following the original work of Garg et al (as well as subsequent works), the basic obfuscator proceeds in three steps:

- We view the NC^1 circuit to be obfuscated as a *branching program* BP (using Barrington’s Theorem [14])—that is, the program is described by m pairs of matrices $(B_{i,0}, B_{i,1})$, each one labelled with an input bit $\text{inp}(i)$. The program is evaluated as follows: for each $i \in [m]$, we choose one of the two matrices $(B_{i,0}, B_{i,1})$, based on the input. Next, we compute the product of the chosen matrices, and based on the product determine the output—there is a unique “accept” (i.e., output 1) matrix, and a unique “reject” (i.e., output 0) matrix.
- The branching program BP is *randomized* using Kilian’s technique [86] (roughly, each pair of matrices is appropriately multiplied with the same random matrix R while ensuring that the output is the same), and then “randomized” some more—each individual matrix is multiplied by a random *scalar* α . Let us refer to this step as *Rand*.
- Finally the randomized matrices are encoded using multilinear encodings with the sets selected appropriately. We here rely on a (simple version) of the *straddling set* idea of [9] to determine the sets. We refer to this step as *Encode*.

(The original construction as well as the subsequent works also consisted of several other steps, but for our purposes these will not be needed.) The obfuscated program is now evaluated by using the multilinear operations to evaluate the branching program and finally appropriately use the zero-test to determine the output of the program.

Roughly speaking, the idea behind the basic obfuscator is that the multilinear encodings *intuitively* ensure that any attacker getting the encoding needs to multiply matrices along paths that corresponds to some input to the branching program (the straddling sets are used to ensure that the input is used consistently in the evaluation)¹¹; the scalars α , roughly speaking, ensure that a potential attacker without loss of generality can use a *single* “multiplication-path” and still succeed with roughly the same probability, and finally, Kilian’s randomization steps ensures that if an attacker *only* operates on matrices along a single path that corresponds to some input x (in a consistent way), then its output can be perfectly simulated given just the output of the circuit on input x . (The final step relies on the fact that the output of the circuit uniquely determines product of the branching program along the path, and Kilian’s randomization then ensures that the matrices along the path are random conditioned on the product being this unique value.) Thus, if an attacker can tell apart obfuscations of two programs BP_0, BP_1 , there must exist some input on which they produce different outputs. The above intuitions can indeed be formalized w.r.t. *generic attackers* (that only operate on the encodings in a legal way, respecting the set restrictions), relying on arguments from [36, 9]. This already suffices to prove that the basic obfuscator is an indistinguishability obfuscator assuming the encodings are *multi-message* semantically secure.¹²

The Merge Procedure To base security on the weaker assumption of (constant-message) semantical security, we will add an additional program transforma-

¹¹The encodings, however, still permit an attacker to add elements within matrices.

¹²As mentioned above, there are still some minor subtleties involved in doing this: the analyses of [36, 9] implicitly show that all *polynomial-size* legal arithmetic circuits are constant with overwhelming probability, but by slightly tweaking the constructions and the analyses to ensure a “perfect” simulation property, we can extend these arguments to hold against *all* (arbitrary-size) legal arithmetic circuits and thus base security on multi-message semantical security.

tion steps before the Rand and Encode steps. Roughly speaking, we would like to have a method $\text{Merge}(BP_0, BP_1, b)$ that “merges” BP_0 and BP_1 into a single branching program that evaluates BP_b ; additionally, we require that $\text{Merge}(BP_0, BP_1, 0)$ and $\text{Merge}(BP_0, BP_1, 1)$ only differ in a constant number of matrices. We achieve this merge procedure by connecting together BP_0, BP_1 into a branching program of double width and adding two “switch” matrices in the beginning and the end, determining if we should go “up” or “down”. Thus, to switch between $\text{Merge}(BP_0, BP_1, 0)$ (which is functionally equivalent to BP_0) and $\text{Merge}(BP_0, BP_1, 1)$ (which is functionally equivalent to BP_1) we just need to switch the “switch matrices”. More precisely, given branching programs BP_0 and BP_1 described respectively by pairs of matrices $\{(B_{i,0}^0, B_{i,1}^0), (B_{i,0}^1, B_{i,1}^1)\}_{i \in [m]}$, we construct a merged program $\text{Merge}(BP_0, BP_1, b)$ described by $\{(\hat{B}_{i,0}^0, \hat{B}_{i,1}^0)\}_{i \in [m+2]}$ such that

$$\hat{B}_{i,b}^0 = \hat{B}_{i,b}^1 = \begin{pmatrix} B_{(i-1),b}^0 & 0 \\ 0 & B_{(i-1),b}^1 \end{pmatrix} \text{ for all } 2 \leq i \leq m+1 \text{ and } b \in \{0, 1\}$$

and the first and last matrices are given by:

$$\begin{aligned} \hat{B}_{1,b}^0 &= \hat{B}_{m+2,b}^0 = I_{2w \times 2w} && \text{for } b \in \{0, 1\} \\ \hat{B}_{1,b}^1 &= \hat{B}_{m+2,b}^1 = \begin{pmatrix} 0 & I_{w \times w} \\ I_{w \times w} & 0 \end{pmatrix} && \text{for } b \in \{0, 1\} \end{aligned}$$

It directly follows from the construction that $\text{Merge}(BP_0, BP_1, 0)$ and $\text{Merge}(BP_0, BP_1, 1)$ differ only in the first and the last matrices (i.e., the “switch” matrices). Furthermore, it is not hard to see that $\text{Merge}(BP_0, BP_1, b)$ is functionally equivalent to BP_b .

Our candidate obfuscator is now defined as $iO(B) = \text{Encode}(\text{Rand}(\text{Merge}(BP, I, 0)))$, where I is simply a “dummy” program of the same size as BP .¹³

¹³This description oversimplifies a bit. Formally, the Rand step needs to depend on the field

The idea behind the merge procedure is that to prove that obfuscations of two programs BP_0, BP_1 are indistinguishable, we can come up with a sequence of hybrid experiments that start with $iO(BP_0)$ and end with $iO(BP_1)$, but between any two hybrids only changes a constant number of encodings, and thus we may rely on semantic security of multilinear encodings to formalize the above intuitions. At a high level, our strategy will be to matrix-by-matrix, replace the dummy branching program in the obfuscation of BP_0 with the branching program for BP_1 . Once the entire dummy branching program has been replaced by BP_1 , we flip the “switch” so that the composite branching program now computes the branching program for BP_1 . We then replace the branching program for BP_0 with BP_1 , matrix by matrix, so that we have two copies of the branching program for BP_1 . We now flip the “switch” again, and finally restore the dummy branching program, so that we end up with one copy of BP_1 and one copy of the dummy, which is now a valid obfuscation of BP_1 . In this way, we transition from an obfuscation of BP_0 to an obfuscation of BP_1 , while only changing a small piece of the obfuscation in each step. (On a very high-level, this approach is somewhat reminiscent of the Naor-Yung “two-key” approach in the context of CCA security [102] and the “two-key” bootstrapping result for indistinguishability obfuscation due to Garg et al [57]—in all these approaches the length of the scheme is artificially doubled to facilitate a hybrid argument. It is perhaps even more reminiscent of the Feige-Shamir “trapdoor witness” approach for constructing zero-knowledge arguments [53], whereby an additional “dummy” trapdoor witness is introduced in the construction to enable the security proof.)

More precisely, consider the following sequence of hybrids.

size used in the Encode steps, and thus in our formal treatment we combine these two steps together.

- We start off with $iO(BP_0) = \text{Enc}(\text{Rand}(\text{Merge}(BP_0, I, 0)))$
- We consider a sequence of hybrids where we gradually change the dummy program I to become BP_1 ; that is, we consider $\text{Encode}(\text{Rand}(\text{Merge}(BP_0, BP', 0)))$, where BP' is “step-wise” being populated with elements from BP_1 .
- We reach $\text{Encode}(\text{Rand}(\text{Merge}(BP_0, BP_1, 0)))$.
- We turn the “switch” : $\text{Encode}(\text{Rand}(\text{Merge}(BP_0, BP_1, 1)))$.
- We consider a sequence of hybrids where we gradually change the BP_0 to become BP_1 ; that is, we consider $\text{Encode}(\text{Rand}(\text{Merge}(BP', BP_1, 1)))$, where BP' is “step-wise” being populated with elements from BP_1 .
- We reach $\text{Encode}(\text{Rand}(\text{Merge}(BP_1, BP_1, 1)))$.
- We turn the “switch” back: $\text{Encode}(\text{Rand}(\text{Merge}(BP_1, BP_1, 0)))$.
- We consider a sequence of hybrids where we gradually change the second BP_1 to become I ; that is, we consider $\text{Encode}(\text{Rand}(\text{Merge}(BP_1, BP', 0)))$, where BP' is “step-wise” being populated with elements from I .
- We reach $\text{Encode}(\text{Rand}(\text{Merge}(BP_1, I, 0))) = iO(BP_1)$.

By construction we have that if BP_0 and BP_1 are functionally equivalent, then so will all the hybrid programs—the key point is that we only “morph” between two branching programs on the “inactive” part of the merged branching program. Furthermore, by construction, between any two hybrids we only change a constant number of elements. Thus, if some distinguisher can tell apart $iO(BP_0)$ and $iO(BP_1)$, it must be able to tell apart two consecutive hybrids. But, by semantic security it then follows that some “legal” arithmetic circuit can tell apart the encodings in the two hybrids. Roughly speaking, we can now rely on simulation

security of the basic obfuscator w.r.t. to just *legal* arithmetic circuits to complete the argument. A bit more precisely, based on BP_0 , BP_1 and the hybrid index i , we can define a message distribution D_{i,BP_0,BP_1} that is valid (by the simulation arguments in [9]) as long as BP_0 is functionally equivalent to BP_1 , yet our distinguisher manages to distinguish messages samples from D_{i,BP_0,BP_1} , contradicting semantical security.

Dealing with branching programs with non-unique outputs There is a catch with the final step though. Recall that to rely on Kilian’s simulation argument it was crucial that there are *unique* accept and reject matrices. For our “merged” programs, this is no longer the case (the output matrix is also a function of the second “dummy” program), and thus it is no longer clear how to prove that the message distribution above is valid. We overcome this issue by noting that the *first column* of the output matrix actually is unique, and this is all we need to determine the output of the branching program; we refer to such branching programs as *fixed output-column branching programs*. Consequently it suffices to release encodings of the *just* first column (as opposed to the whole matrices) of the last matrix pair in the branching program, and we can still determine the output of the branching program. As we show, for such a modified scheme, we can also simulate the (randomized) matrices along an “input-path” given just the first column of the output matrix.

A Modular Analysis: Neighboring-Matrix Indistinguishability Obfuscation

In the actual proof, we provide a more modular analysis of the above two steps (that may be interesting in its own right).

- We define a notion of *neighboring-matrix indistinguishability obfuscation*,

which relaxes indistinguishability obfuscation by only requiring security to hold w.r.t. any two functionally equivalent branching programs that differ in at most a constant number of matrices.

- We then use the above merge procedure (and the above hybrid argument) to show that the existence of a neighboring-matrix iO for all “fixed output column” branching programs implies the existence of a “full-fledged” iO .
- We finally use the “basic obfuscator” construction to show how to construct a neighboring-matrix iO for all fixed output column branching programs based on (constant-message) semantical security.

Basing Security on a (Single) Falsifiable Assumption To base security on a falsifiable assumption, we rely on a different merge procedure from the work of Boyle, Chung and Pass [32]: Given two NC_1 circuits C_0, C_1 taking (at most) n -bit inputs, and a string z , let $\widehat{\text{Merge}}(C_0, C_1, z)$ be a circuit that on input x runs $C_0(x)$ if $x \geq z$ and $C_1(x)$ otherwise; in essence, this procedure lets us “traverse” between C_0 and C_1 while provably only changing the functionality on at most one input. ([32] use this type of merged circuits to perform a binary search and prove that indistinguishability obfuscation implies differing-input obfuscation for circuits that differ in only polynomially many inputs.) We now define a notion of *neighboring-input iO* , which relaxes iO by only requiring that security holds with respect to “neighboring-input” programs $\widehat{\text{Merge}}(C_0, C_1, z), \widehat{\text{Merge}}(C_0, C_1, z+1)$ that are functionally equivalent. Note that checking whether $\widehat{\text{Merge}}(C_0, C_1, z), \widehat{\text{Merge}}(C_0, C_1, z+1)$ are functionally equivalent is easy: they are equivalent iff $C_0(z) = C_1(z)$. (As such, the assumption that a scheme satisfies neighboring-input iO is already an efficiently falsifiable assumption.) Furthermore, by a simple hybrid argument over $z \in \{0, 1\}^n$, *exponentially-secure* neighboring-input iO

implies “full” $i\mathcal{O}$ —exponential security is needed since we have 2^n hybrids. (We mention a very recent work by Gentry, Lewko and Waters [65] in the context of *witness encryption* [59] that similarly defines a falsifiable primitive “positional witness encryption” that implies the full-fledged notion with an exponential security loss.)

Additionally, note that to show that our construction satisfies exponentially-secure neighboring-input $i\mathcal{O}$, we only need to rely on exponentially-secure semantical security w.r.t. classes of sets and message distributions corresponding to programs of the form $\widehat{\text{Merge}}(C_0, C_1, z)$, $\widehat{\text{Merge}}(C_0, C_1, z + 1)$. Equivalently, it suffices to rely on exponentially-secure semantical security w.r.t. a *single* distribution over sets and message distributions corresponding to uniformly selected programs $\widehat{\text{Merge}}(C_0, C_1, z)$, $\widehat{\text{Merge}}(C_0, C_1, z + 1)$ (i.e., z, C_0, C_1 are picked at random); again, this only results in an exponential security loss. Finally, by padding the security parameter of the multilinear encodings in the construction, it actually suffices to rely on subexponential security.

3.1.5 Discussion and Future Work

We have introduced a new security notion, *semantical security*, for multilinear (a.k.a. graded) encodings, which captures a general (but quite restrictive) *class* of algebraic decisional assumption over multilinear encodings. Our main result demonstrates the existence of indistinguishability obfuscators ($i\mathcal{O}$) assuming the existence of semantically secure multilinear encodings and the LWE assumption; as far as we know, this yields the first construction of $i\mathcal{O}$ based on a “simple-to-state” assumption about some algebraic primitive (namely, multilin-

ear encodings) for which candidate constructions are known.

We additionally show that it suffices to assume the existence of encoding schemes that satisfy a *specific, falsifiable, instance* of semantical security (i.e., that a specific assumption in the class holds w.r.t. the encoding scheme); this time, however, we need to assume *subexponentially-hard* semantical security. This shows that under subexponential reductions, indistinguishability obfuscation can be based on a single, non-interactive and falsifiable, assumption.

We finally consider various strengthenings of semantical security, which (among other things) motivate why in our definition of semantical security, we restrict the class of algebraic decisional assumptions: we show that the assumption that “every non-interactive algebraic decisional assumption that holds against generic attackers holds against nuPPT attackers” is false.

Our work leaves open several interesting questions:

- Can we base iO on *polynomial-hardness* of a falsifiable (and preferably non-interactive) assumption (using a security-preserving reduction)? Note that for many *applications* of iO (e.g., functional encryption [57]) it suffices to require indistinguishability for restricted distributions of programs that (with overwhelming probability) are *provably* functionally equivalent; for these applications, our proof already shows they can be based on specific, falsifiable, instances of semantical security (without assuming subexponential hardness).
- Even in the regime of subexponential hardness, the specific assumption that we use—although it is a special case of semantical security—is not particularly natural, and does not have a particularly “simple” descrip-

tion. In essence, we consider semantical security with respect to distributions over elements that describe the obfuscation of a random branching program. (As such, in our eyes, perhaps the best reason to believe this assumption is true that it is a falsifiable special case of semantical security). It would be much more desirable to base security on semantical security w.r.t. a single *simple* and *natural* distribution over $\vec{m}_0, \vec{m}_1, \vec{z}$, where, for instance, similar to the GDDH assumption, \vec{z} are uniformly random elements. We conjecture that our assumption actually can be “massaged” into a nicer looking assumption, closer in spirit to the GDDH assumption. Two recent works take a major step in this direction. The elegant work of Gentry, Lewko and Waters [65]¹⁴ bases *witness encryption* [59] on exponential hardness of some simple assumptions over multilinear encodings—the “multilinear subgroup eliminations assumption” and the “multilinear subgroup decision assumption” (which are closer in spirit to the GDDH assumption); however, in contrast to our work they rely on multilinear (graded) encodings over composite-order rings (for which the only candidate is a modified variant of [49]), or require more complex assumptions over prime-order rings (that still are false for the [54] construction); furthermore, they require several additional functionalities from graded encodings—in particular, “subring generation”, and “subring sampling”, which require releasing additional “auxiliary elements” and thus challenges security (which is why a variant of the [49] construction is needed). Even more recently, the beautiful work by Gentry, Lewko, Sahai and Waters [64] manages to demonstrate also *iO* from just the multilinear sub-

¹⁴This result is subsequent to our results on *iO* from (entropic) semantical security, but precedes our results on *iO* from single-distribution semantical security.

group assumption over composite-order rings.¹⁵ Just as [65] they require the additional functionalities from graded encoding scheme (and as such the only candidate construction currently know is the variant of the [49] scheme introduced in [65]). Although the implementation details are quite different, the construction in [64] follows our general approach of “merging” threads of branching programs (we here consider only two threads, whereas they consider multiple), and using a switch between “active” and “inactivate” threads. (Additionally, their notion of a “positional” iO is closely related to our notion of neighboring-input iO .)¹⁶

- Another interesting question is finding other applications of entropic semantically secure multilinear encodings. Our impossibility results—which show that there exist algebraic decisional assumptions that are false despite being true w.r.t. generic attackers—present a further challenge to the practice of arguing the plausibility of an assumption (even a “DDH-type” assumption) through security arguments in the generic model. At this point it seems that checking whether some specific algebraic assumption falls within the class of assumptions considered by entropic semantical security (or perhaps even just uber security) may be a viable replacement to the standard “sanity check” of arguing security in the generic model.
- In this paper we have focused on indistinguishability obfuscation. An interesting problem is basing stronger notions of obfuscation on some succinct and natural assumption on a low-level primitive. We mention

¹⁵This result is subsequent to our results on iO from (entropic) semantical security, and appears to be concurrent to (appearing on e-Print only a few days after) our results on iO from single-distribution semantical security.

¹⁶But as mentioned above, the results relying on neighboring-input iO were not part of our original manuscript and appear to be concurrent to the ones in [64].

that our result that any scheme satisfying $i\mathcal{O}$ security w.r.t. “neighboring-matrix” programs can be turned in a “fully” secure scheme, applies also to differing-input security.

A recent beautiful work of Bitansky, Canetti, Kalai and Paneth [17] introduces a strengthening of semantical security (called “strong-sampler” semantical security) which also consider non-samplable (i.e., computationally unbounded) message distributions (as opposed to nuPPT distributions as we consider here); their key result demonstrated the existence of VGB (virtual grey-box secure) [16] obfuscators for NC^1 assuming strong-sampler semantical security. VGB security is a strengthening of $i\mathcal{O}$; but it is not known how to bootstrap VGB for NC^1 to all polynomial-size circuits.

3.1.6 Outline of the Paper

We provide some preliminaries in Section 3.2. We define semantical security of multilinear (aka graded) encodings in Section 3.3. Our construction of an indistinguishability obfuscator and its proof of security is provided in Section 3.4. We show how to slightly modify the construction to be based on a single (falsifiable) instance of semantical security in Section 3.5. We finally study alternative notions of security for multilinear encodings in Section 3.6.

3.2 Preliminaries

Let \mathcal{N} denote the set of positive integers, and $[n]$ denote the set $\{1, 2, \dots, n\}$. Let \mathcal{Z} denote the integers, and \mathbb{Z}_p the integers modulo p . Given a string x , we let

$x[i]$, or equivalently x_i , denote the i -th bit of x . For a matrix M , we let $M[i, j]$ denote the entry of M in the i th row and j th column. We use \mathbf{e}_k to denote the vector that is 1 in position k , and 0 in all other positions. The length of \mathbf{e}_k is generally clear from the context. We use $I_{w \times w}$ to denote the identity matrix with dimension $w \times w$.

By a probabilistic algorithm we mean a Turing machine that receives an auxiliary random tape as input. If M is a probabilistic algorithm, then for any input x , $M(x)$ represents the distribution of outputs of $M(x)$ when the random tape is chosen uniformly. $M(x; r)$ denotes the output of M on input x when the random tape is fixed to r . An oracle algorithm M^O is a machine M that gets oracle access to another machine O , that is, it can access O 's functionality as a black-box.

By $x \leftarrow S$, we denote an element x is sampled from a distribution S . If F is a finite set, then $x \leftarrow F$ means x is sampled uniformly from the set F . To denote the ordered sequence in which the experiments happen we use semicolon, e.g. $(x \leftarrow S; (y, z) \leftarrow A(x))$. Using this notation we can describe probability of events. For example, if $p(\cdot, \cdot)$ denotes a predicate, then $\Pr[x \leftarrow S; (y, z) \leftarrow A(x) : p(y, z)]$ is the probability that the predicate $p(y, z)$ is true in the ordered sequence of experiments $(x \leftarrow S; (y, z) \leftarrow A(x))$. The notation $\{(x \leftarrow S; (y, z) \leftarrow A(x) : (y, z))\}$ denotes the resulting probability distribution $\{(y, z)\}$ generated by the ordered sequence of experiments $(x \leftarrow S; (y, z) \leftarrow A(x))$. We define the support of a distribution $\text{supp}(S)$ to be $\{y : \Pr[x \leftarrow S : x = y] > 0\}$.

By isZero , we denote the predicate such that $\text{isZero}(x) = 1$ exactly when $x = 0$, and $\text{isZero}(x) = 0$ otherwise.

3.2.1 Obfuscation

We recall the definition of indistinguishability obfuscation due to [12].

Definition 24 (Indistinguishability Obfuscator). *A uniform PPT machine iO is an indistinguishability obfuscator for a class of circuits $\{C_n\}_{n \in \mathbb{N}}$ if the following conditions are satisfied*

- **Correctness:** *There exists a negligible function ε such that for every $n \in \mathbb{N}$, for all $C \in C_n$, we have*

$$\Pr[C' \leftarrow iO(1^n, C) : \forall x, C'(x) = C(x)] \geq 1 - \varepsilon(n)$$

- **Security:** *For every pair of circuit ensembles $\{C_n^0\}_{n \in \mathbb{N}}$ and $\{C_n^1\}_{n \in \mathbb{N}}$ such that for all $n \in \mathbb{N}$, for every pair of circuits $C_n^0, C_n^1 \in C_n$ such that $C_n^0(x) = C_n^1(x)$ for all x the following holds: For every nuPPT adversary A there exists a negligible function ε such that for all $n \in \mathbb{N}$,*

$$|\Pr[C' \leftarrow iO(1^n, C_n^0) : A(1^n, C') = 1] - \Pr[C' \leftarrow iO(1^n, C_n^1) : A(1^n, C') = 1]| \leq \varepsilon(n)$$

We additionally say that iO is subexponentially-secure if there exists some constant $\alpha > 0$ such that for every nuPPT A the above indistinguishability gap is bounded by $\varepsilon(n) = 2^{-O(n^\alpha)}$.

Note: We observe that the above definition allows for a negligible correctness error. That is, for any circuit C , there is a negligible fraction of “bad” randomness r such that $iO(C; r)$ is not functionally equivalent to C . However, if we can efficiently check if r is “bad”, we can modify iO so that $iO(C; r)$ outputs C in the clear if r is “bad”. Then the modified iO has perfect correctness, and its security remains intact since only a negligible fraction of r are “bad”. We note that our

construction, as well as all previous ones, have the property that a “bad” r can be efficiently detected, and thus these schemes can be modified to have perfect correctness.

We now recall the definitions of $i\mathcal{O}$ for NC^1 and P/poly .

Definition 25 (Indistinguishability Obfuscator for NC^1). *A uniform PPT machine $i\mathcal{O}$ is an indistinguishability obfuscator for NC^1 if for every constant c , $i\mathcal{O}(c, \cdot, \cdot)$ is an indistinguishability obfuscator for the class of circuits $C^c = \{C_n^c\}_{n \in \mathbb{N}}$ where C_n^c is the set of circuits that have size at most n^c , and have depth at most $c \log n$.*

Definition 26 (Indistinguishability Obfuscator for P/poly). *A uniform PPT machine $i\mathcal{O}$ is an indistinguishability obfuscator for P/poly if for every constant c , $i\mathcal{O}(c, \cdot, \cdot)$ is an indistinguishability obfuscator for the class of circuits $\mathcal{P}^c = \{\mathcal{P}_n^c\}_{n \in \mathbb{N}}$ where \mathcal{P}_n^c is the set of circuits that have size at most n^c .*

The following simple lemma will be useful in the sequel.

Lemma 19. *Let $i\mathcal{O}$ be a (subexponentially-secure) indistinguishability obfuscator for C^1 . Then $i\mathcal{O}'$ defined as $i\mathcal{O}'(c, 1^n, C) = i\mathcal{O}(1^{n^c}, C)$ is a (subexponentially-secure) indistinguishability obfuscator for NC^1 .*

Proof. Consider any pair of circuit ensembles $\{C_n^0\}_{n \in \mathbb{N}}, \{C_n^1\}_{n \in \mathbb{N}}$ in C^c . Assume for contradiction that there exists some nuPPT A and a polynomial $p(\cdot)$ such that $A(1^n)$ distinguishes $i\mathcal{O}'(c, 1^n, C_n^0) = i\mathcal{O}(1^{n^c}, C_n^0)$ and $i\mathcal{O}'(c, 1^n, C_n^1) = i\mathcal{O}(1^{n^c}, C_n^1)$ with probability $1/p(n)$ for infinitely many n . Note that for every n , $C_n^0, C_n^1 \in C_{n^c}^1$. Thus, for infinitely many $n \in \mathbb{N}$, there exists circuits $C_n^0, C_n^1 \in C_{n^c}^1$ such that $A(1^n)$ distinguishes $i\mathcal{O}(1^{n^c}, C_n^0)$ and $i\mathcal{O}(1^{n^c}, C_n^1)$ with probability $1/p(n)$. In other words, for infinitely many $n' \in \mathbb{N}$ of the form $n' = n^c$, there exist circuits $\tilde{C}_{n'}^0 = C_n^0, \tilde{C}_{n'}^1 = C_n^1$

such that the nuPPT $A'(1^{n'}) = A(1^n)$ distinguishes $i\mathcal{O}(1^{n'}, \tilde{C}_{n'}^0)$ and $i\mathcal{O}(1^{n'}, \tilde{C}_{n'}^1)$ with probability $1/p(n) = 1/p(n'^{1/c})$, which contradicts that $i\mathcal{O}$ is an indistinguishability obfuscator for C^1 .

The same argument also works in the context of subexponential security. \square

3.2.2 Branching programs for NC^1

We recall the notion of a branching program.

Definition 27 (Matrix Branching Program). A branching program of width w and length m for n -bit inputs is given by a sequence:

$$BP = \{\text{inp}(i), B_{i,0}, B_{i,1}\}_{i=1}^m$$

where each $B_{i,b}$ is a permutation matrix in $\{0, 1\}^{w \times w}$, $\text{inp}(i) \in [n]$ is the input bit position examined in step i .

Then the output of the branching program on input $x \in \{0, 1\}^n$ is as follows:

$$BP(x) \stackrel{\text{def}}{=} \begin{cases} 1, & \text{if } (\prod_{i=1}^m B_{i,x[\text{inp}(i)]}) \cdot \mathbf{e}_1 = \mathbf{e}_1. \\ 0, & \text{otherwise} \end{cases}$$

The branching program is said to be oblivious if $\text{inp} : [m] \rightarrow [n]$ is a fixed function, independent of the function being evaluated.

The above definition differs slightly from the definition of matrix branching programs generally used, which have the slightly stronger requirement that

$\prod_{i=1}^n B_{i,x[\text{inp}(i)]} = I_{w \times w}$ when $BP(x)$ is accepting, and $\prod_{i=1}^n B_{i,x[\text{inp}(i)]} = P_{\text{reject}}$ for some fixed permutation matrix $P_{\text{reject}} \neq I_{w \times w}$ when $BP(x)$ is rejecting. More generally,

Definition 28. *The branching program is said to have fixed accept and reject matrices P_{accept} and P_{reject} if, for all $x \in \{0, 1\}^n$,*

$$\prod_{i=1}^m B_{i,x[\text{inp}(i)]} = \begin{cases} P_{\text{accept}} & \text{when } BP(x) = 1 \\ P_{\text{reject}} & \text{when } BP(x) = 0 \end{cases}$$

We now have the following theorem due to Barrington:

Theorem 20. ([14]) *There exist 5×5 permutation matrices P_{accept} and P_{reject} with $P_{\text{accept}} \cdot \mathbf{e}_1 = \mathbf{e}_1$, and $P_{\text{reject}} \cdot \mathbf{e}_1 = \mathbf{e}_k$ where $k \neq 1$ such that the following holds. For any depth d and input length n , there exists a length $m = 4^d$, a labeling function $\text{inp} : [m] \rightarrow [n]$ such that, for every fan-in 2 boolean circuit C of depth d and n input bits, there exists an oblivious matrix branching program $BP = \{\text{inp}(i), B_{i,0}, B_{i,1}\}_{i=1}^m$, of width 5 and length m that computes the same function as the circuit C .*

In particular, every circuit in NC^1 has a polynomial length branching program of width 5. Further, two circuits of the same depth d will have the same fixed accepting and rejecting permutations P_{accept} and P_{reject} , and a fixed labelling function $\text{inp} : [m] \rightarrow [n]$.

The branching programs we consider in this work will not have fixed output matrices. However, the first column of their output matrices will be fixed and depend only on the output of the program. That is, the first column of the output matrix is either $\mathbf{p}_{\text{accept}}$ or $\mathbf{p}_{\text{reject}}$, depending on whether the program accepts or rejects. Furthermore, we will consider ensembles of classes of programs where these fixed columns $\mathbf{p}_{\text{accept}}$ and $\mathbf{p}_{\text{reject}}$ are the same for all programs in every class in the ensemble.

Definition 29 (Fixed Output Column Ensemble). *An ensemble of classes of branching programs $\mathcal{B} = \{\mathcal{B}_n\}_{n \in \mathcal{N}}$ where \mathcal{B}_n contains branching programs of constant width w , is a fixed output column ensemble if there exists vectors $\mathbf{p}_{\text{accept}}, \mathbf{p}_{\text{reject}} \in \{0, 1\}^w$ such that for every $n \in \mathcal{N}$, every branching program $BP = \{\text{inp}(i), B_{i,0}, B_{i,1}\}_{i=1}^m \in \mathcal{B}_n$ and every input x it holds that*

$$\left(\prod_{i=1}^m B_{i,x[\text{inp}(i)]} \right) \cdot \mathbf{e}_1 = \begin{cases} \mathbf{p}_{\text{accept}} & \text{when } BP(x) = 1 \\ \mathbf{p}_{\text{reject}} & \text{when } BP(x) = 0 \end{cases}$$

Subsequently, whenever we refer to an ensemble of classes of branching programs we will implicitly be referring to a fixed output column ensemble.

3.3 Semantically Secure Graded Encoding Schemes

In this section we define what it means for a graded encoding scheme to be semantically secure. We start by recalling the notion of graded encoding schemes due to Garg, Gentry and Halevi [54].

3.3.1 Graded Encoding Schemes

Graded (multilinear) encoding schemes were originally introduced in the work of Garg, Gentry and Halevi [54]. Just as [36, 9], we here rely on “set-based” (or “asymmetric”) graded encoding; these were originally called “generalized” graded encodings in [54]. Following [57, 9] and the notion of “multilinear jigsaw puzzles” from [57], we additionally enable anyone with the secret parameter to encode *any* elements (as opposed to just *random* elements as in [54]).

Definition 30 ((k, R) -Graded Encoding Scheme). A (k, R) -graded encoding scheme for $k \in \mathbb{N}$ and ring R is a collection of sets $\{E_S^\alpha : \alpha \in R, S \subseteq [k]\}$ with the following properties

- For every $S \subseteq [k]$ the sets $\{E_S^\alpha : \alpha \in R\}$ are disjoint.
- There are associative binary operations \oplus and \ominus such that for every $\alpha_1, \alpha_2 \in R$, $S \subseteq [k]$, $u_1 \in E_S^{\alpha_1}$ and $u_2 \in E_S^{\alpha_2}$ it holds that $u_1 \oplus u_2 \in E_S^{\alpha_1 + \alpha_2}$ and $u_1 \ominus u_2 \in E_S^{\alpha_1 - \alpha_2}$ where ‘+’ and ‘-’ are the addition and subtraction operations in R .
- There is an associative binary operation \otimes such that for every $\alpha_1, \alpha_2 \in R$, $S_1, S_2 \subseteq [k]$ such that $S_1 \cap S_2 = \emptyset$, $u_1 \in E_{S_1}^{\alpha_1}$ and $u_2 \in E_{S_2}^{\alpha_2}$ it holds that $u_1 \otimes u_2 \in E_{S_1 \cup S_2}^{\alpha_1 \cdot \alpha_2}$ where ‘.’ is multiplication in R .

Definition 31 (Graded Encoded Scheme). A graded encoding scheme \mathcal{E} is associated with a tuple of PPT algorithms, $(\text{InstGen}_\mathcal{E}, \text{Enc}_\mathcal{E}, \text{Add}_\mathcal{E}, \text{Sub}_\mathcal{E}, \text{Mult}_\mathcal{E}, \text{isZero}_\mathcal{E})$ which behave as follows:

- **Instance Generation:** $\text{InstGen}_\mathcal{E}$ takes as input the security parameter 1^n and multilinearity parameter 1^k , and outputs secret parameters sp and public parameters pp which describe a (k, R) -graded encoding scheme $\{E_S^\alpha : \alpha \in R, S \subseteq [k]\}$. We refer to E_S^α as the set of encodings of the pair (α, S) . We restrict to graded encoding schemes where R is \mathbb{Z}_p and p is a prime exponential in n and k .
- **Encoding:** $\text{Enc}_\mathcal{E}$ takes as input the secret parameters sp , an element $\alpha \in R$ and set $S \subseteq [k]$, and outputs a random encoding of the pair (α, S) .
- **Addition:** $\text{Add}_\mathcal{E}$ takes as input the public parameters pp and encodings $u_1 \in E_{S_1}^{\alpha_1}, u_2 \in E_{S_2}^{\alpha_2}$, and outputs an encoding of the pair $(\alpha_1 + \alpha_2, S)$ if $S_1 = S_2 = S$ and outputs \perp otherwise.

- *Negation:* $\text{Sub}_{\mathcal{E}}$ takes as input the public parameters pp and encodings $u_1 \in E_{S_1}^{\alpha_1}, u_2 \in E_{S_2}^{\alpha_2}$, and outputs an encoding of the pair $(\alpha_1 - \alpha_2, S)$ if $S_1 = S_2 = S$ and outputs \perp otherwise.
- *Multiplication:* $\text{Mult}_{\mathcal{E}}$ takes as input the the public parameters pp and encodings $u_1 \in E_{S_1}^{\alpha_1}, u_2 \in E_{S_2}^{\alpha_2}$, and outputs an encoding of the pair $(\alpha_1 \cdot \alpha_2, S_1 \cup S_2)$ if $S_1 \cap S_2 = \emptyset$ and outputs \perp otherwise.
- *Zero testing:* $\text{isZero}_{\mathcal{E}}$ takes as input the public parameters pp and an encoding $u \in E_S(\alpha)$, and outputs 1 if and only if $\alpha = 0$ and S is the universe set $[k]$.¹⁷

Whenever it is clear from the context, to simplify notation we drop the subscript \mathcal{E} when we refer to the above procedures (and simply call them $\text{InstGen}, \text{Enc}, \dots$).

In known candidate constructions [54, 49], encodings are “noisy” and the noise level increases with each operation; the parameters, however, are set so that any $\text{poly}(n, k)$ operations can be performed without running into trouble. For convenience of notation (and just like all other works in the area), we ignore this noise issue.¹⁸

Note that the above procedures allow algebraic operations on the encodings in a restricted way. Given the public parameters and encodings made under the sets \vec{S} , one can only perform algebraic operations that are allowed by the

¹⁷In the candidate scheme given by [54], isZero may not have perfect correctness: the generated instances (pp, sp) can be “bad” with some negligible probability, so that there could exist an encoding u of a nonzero element where $\text{isZero}(\text{pp}, u) = 1$. However, these “bad” parameters can be efficiently detected during the execution of InstGen . We can thus modify the encoding scheme to simply set $\text{Enc}(\text{pp}, e) = e$ whenever the parameters are “bad” (and appropriately modify $\text{Add}, \text{Sub}, \text{Mult}$ and isZero so that they operate on “unencoded” elements. This change ensures that, for every pp , including “bad” ones, the zero test procedure isZero works with perfect correctness. We note that since bad parameters occur only with negligible probability, this change does not affect the security of the encodings.

¹⁸The above definition can be easily generalized to deal with the candidates by only requiring that the above conditions hold when u_1, u_2 have been obtained by $\text{poly}(n, k)$ operations.

structure of the sets in \vec{S} . We call such operations \vec{S} -respecting and formalize this notion as follows:

Definition 32 (Set Respecting Arithmetic Circuits). *For any sequence \vec{S} of subsets of $[k]$, we say that an arithmetic circuit C (i.e. gates perform only ring operations $\{+, -, \cdot\}$) is \vec{S} -respecting if it holds that*

- *Every input wire of C is tagged with some set in \vec{S} .*
- *For every $+$ and $-$ gate in C , if the tags of the two input wires are the same set S then the output wire of the gate is tagged with S . Otherwise the output wire is tagged with \perp .*
- *For every \cdot gate in C , if the tags of the two input wires are sets S_1 and S_2 and $S_1 \cap S_2 = \emptyset$ then the output wire of the gate is tagged with $S_1 \cup S_2$. Otherwise the output wire is tagged with \perp .*
- *It holds that the output wire is tagged with the universe set $[k]$.¹⁹*

We say that a circuit C is weakly \vec{S} -respecting if all the above conditions hold except the last, that is, the output wire may be tagged with some set $T \subseteq [k]$, where T is not necessarily equal to $[k]$. We say that C is non terminal \vec{S} -respecting if T is a strict subset of $[k]$.

3.3.2 Semantical Security

We now turn to defining semantical security of graded encoding schemes. Towards explaining our notion of semantical security, let us first consider a “DDH-type” assumption for (asymmetric) multilinear encodings, similar in spirit to

¹⁹For ease of notation, we assume that the description of a set S also contains a description of the universe set $[k]$.

the “graded DDH” assumption of Garg et al [54] (which was in the context of symmetric multilinear encodings, whereas we here consider asymmetric ones). Consider a distribution D sampling n random elements \vec{z} , and let $m_0 = \prod_{i \in [n]} z_i$ be the product of the elements in \vec{z} , and $m_1 = z'$ be just a random element. A DDH-type assumption—let us refer to it as the “asymmetric graded DDH assumption (aGDDH)” —would require that encodings of m_0, \vec{z} and m_1, \vec{z} under the sets S, \vec{T} are indistinguishable as long as (a) S is the target set $[k]$, and (b) S is not the disjoint union of the sets in \vec{T} ; that is, the set-restrictions prohibit “legally” multiplying all the elements of \vec{z} and subtracting them from m_0 or m_1 .

Note that for any such sets S, \vec{T} , the particular (joint) distribution D over m_0, m_1, \vec{z} has a nice “zero-knowledge” property w.r.t. generic attacker: for every (S, \vec{T}) -respecting circuit C , $\text{isZero}(C(\cdot))$ is *constant* over $(m_b, \vec{z}), b \in \{0, 1\}$ with overwhelming probability: that is, there exists some bit c such that with overwhelming probability over $m_0, m_1, \vec{z} \leftarrow D$, $\text{isZero}(C(m_b, \vec{z})) = c$ for $b \in \{0, 1\}$, and as (except with negligible probability) no zero-test query leaks *anything* to a generic attacker. To see this, note that any such $\text{isZero}(C(m, \vec{z}))$ function is of the form $\text{isZero}(a \cdot m + p(\vec{z}))$ where $p(\cdot)$ is a polynomial of degree at most $n - 1$. If $a = 0$ and $p(\cdot)$ is the zero-polynomial, then clearly the function evaluates to 1. If either $a = 1$ or $p(\cdot)$ is a non-zero polynomial, then no matter whether $m = m_0$ or $m = m_1$, $\text{isZero}(C(\cdot, \cdot))$ is evaluating a non-zero polynomial of degree at most n at a random point; by the Schwartz-Zippel lemma, with overwhelming probability (proportional to the field size), both these polynomials will evaluate to a non-zero value, and thus the zero-test will output 0.

We refer to any distribution D satisfying the above “zero-knowledge w.r.t.

generic attackers” property as being *valid* w.r.t. S, \vec{T} . We formalize this notion through what we refer to as a (S, \vec{T}) -*respecting message sampler*. As mentioned in the introduction, for our purposes, we need to consider a more general setting where m_0, m_1 , and S are replaced by *constant-length* vectors $\vec{m}_0, \vec{m}_1, \vec{S}$; for generality, we provide a definition that considers arbitrary length vectors of messages.

Definition 33 (Set-Respecting Operations). *Let $\{k_n\}_{n \in \mathcal{N}}$ be an ensemble where $k_n \in \mathcal{N}$. We say $f = \{f_n\}_{n \in \mathcal{N}}$ is an ensemble of set-respecting operations if for every $n \in \mathcal{N}$, and every pair of sequences of sets \vec{S}, \vec{T} over $[k_n]$ we have that $f_n(\vec{S}, \vec{T})$ outputs a (\vec{S}, \vec{T}) -respecting arithmetic circuit.*

Definition 34 (Valid Message Sampler). *Let \mathcal{E} be a graded encoding scheme. We say that a nuPPT M is a valid message sampler if*

- *M on input 1^n and a public parameter $\text{pp} \in \text{InstGen}(1^n, 1^{k_n})$ computes the ring R associated with pp and next based on only $1^n, 1^{k_n}$ and R generates and outputs*
 - *a pair (\vec{S}, \vec{T}) of sequences of sets over $[k_n]$ and*
 - *a pair (\vec{m}_0, \vec{m}_1) of sequences of $|S|$ ring elements and a sequence \vec{z} of $|T|$ ring elements.*
- *There exists a polynomial $Q(\cdot, \cdot)$ such that for every ensemble $\{k_n\}_{n \in \mathcal{N}}$ and ensemble of set-respecting operations $\{f_n\}_{n \in \mathcal{N}}$, for every $n \in \mathcal{N}$ there exists a constant $c \in \{0, 1\}$ such that that for any $b \in \{0, 1\}$,*

$$\Pr[(\vec{m}_0, \vec{m}_1, \vec{z}, \vec{S}, \vec{T}) \leftarrow M(1^n, \text{pp}); C \leftarrow f_n(\vec{S}, \vec{T}) : \text{isZero}(C(\vec{m}_b, \vec{z})) = c] \geq 1 - Q(n, k_n)/|R|.$$

Let us comment that Definition 34 allows the message sampler M to select $\vec{m}_0, \vec{m}_1, \vec{z}$ based on the ring $R = \mathcal{Z}_p$; note that this is needed even to model the

aGDDH assumption (or else we could not define what it means to pick a uniform element in the ring). On the other hand, to make the notion of valid message samplers as restrictive as possible, we prevent the message selection from depending on pp in any other way. Looking ahead, this restriction makes the notion somewhat nicer behaved; see Lemma 21.

We can now define what it means for a graded encoding scheme to be semantically secure. Roughly speaking, we require that encodings of (\vec{m}_0, \vec{z}) and (\vec{m}_1, \vec{z}) under the sets (\vec{S}, \vec{T}) are indistinguishable as long as $(\vec{m}_0, \vec{m}_1, \vec{z})$ is sampled by a message sampler that is valid w.r.t. (\vec{S}, \vec{T}) .

Definition 35 (Semantic Security). *Let \mathcal{E} be a graded encoding scheme and $q(\cdot)$ and $c(\cdot)$ be polynomials. We say a graded encoding scheme \mathcal{E} is (c, q) -semantically secure if for every polynomial $k(\cdot)$, every ensemble $\{(\vec{S}_n, \vec{T}_n)\}_{n \in \mathbb{N}}$ where \vec{S}_n and \vec{T}_n are sequences of subsets of $[k(n)]$ of length $c(k(n))$ and $q(k(n))$ respectively, for every $\{(\vec{S}_n, \vec{T}_n)\}_{n \in \mathbb{N}}$ -respecting message sampler M and every nuPPT adversary A , there exists a negligible function ϵ such that for every security parameter $n \in \mathbb{N}$,*

$$|\Pr[\mathbf{Output}_0(1^n) = 1] - \Pr[\mathbf{Output}_1(1^n) = 1]| \leq \epsilon(n)$$

where $\mathbf{Output}_b(1^n)$ is A 's output in the following game:

- Let $(\text{sp}, \text{pp}) \leftarrow \text{InstGen}(1^n, 1^{k(n)})$.
- Let $\vec{m}_0, \vec{m}_1, \vec{z} \leftarrow M(1^n, \text{pp})$.
- Let $\vec{u}_b \leftarrow \{\text{Enc}(\text{sp}, \vec{m}_0[i], \vec{S}_n[i])\}_{i=1}^{c(k(n))}, \{\text{Enc}(\text{sp}, \vec{z}[i], \vec{T}_n[i])\}_{i=1}^{q(k(n))}$.
- Finally, run $A(1^n, \text{pp}, \vec{u}_b)$.

We say that \mathcal{E} is (constant-message) semantically secure if it is $(O(1), O(k))$ -semantically secure; we say that \mathcal{E} multi-message semantically secure if it is

$(O(k), O(k))$ -semantically secure. We additionally say that \mathcal{E} is subexponentially-hard semantically secure if there exists some constant $\alpha > 0$ such that for every nuPPT A the above indistinguishability gap is bounded by $\varepsilon(n) = 2^{-O(n^\alpha)}$.²⁰

In analogy with the GDDH assumption, our notion of semantical security restricts to the case when the number of elements encoded is $O(k)$.²¹ As the following lemma (whose proof is delegated to Section 3.9) shows, any such encoding scheme can be modified to one that is secure as long as the number of elements in \vec{z} is (a-priori) polynomially bounded.

Lemma 21. *Let c, ϵ be constants and let \mathcal{E} be a (c, k^ϵ) -semantically secure encoding scheme. Then for every polynomial $q(k)$ there exists a $(c, q(k))$ -semantically secure encoding scheme.*

Also, note that our notion of semantical security requires that security holds w.r.t. to *any* polynomial multilinearity parameter $k(\cdot)$; again, this is without loss of generality: Any encoding scheme \mathcal{E} that is semantically secure for any multilinearity parameter $k(n) \leq n$, can be turned into a new scheme \mathcal{E}' that is (full-fledged) semantically secure, by simply letting $\text{InstGen}'(1^n, 1^k) = \text{InstGen}(1^{n+k}, 1^k)$.

Finally, one may also consider a notion of *unbounded semantical security* (that is provably stronger than semantical security)²² which requires that \mathcal{E}

²⁰We could also have considered an even stronger notion where the adversary A is allowed to be of subexponential-size; this will not be needed for our result, but may be useful in other contexts.

²¹This restriction was suggested in [17] and independently by Hoeteck Wee; our original formulation of semantical security considered an unbounded polynomial number of elements in \vec{z} (but our proof of security only relied on security for $O(k)$ elements). We now refer to this stronger notion as *unbounded semantical security*; see below.

²²Any semantically secure encoding scheme \mathcal{E} can be modified into a new encoding scheme \mathcal{E}' that still is semantically secure but not unbounded semantically secure. Simply let each en-

is $(O(1), q(k))$ -semantically secure for *every* polynomial $q(k)$; this notion is not needed for our results. A recent result by [17] shows that for natural *special cases* of message samplers, *unbounded* single-message semantical security implies multi-message semantical security; we mention that this result only applies in the regime of polynomial security (and in particular does not apply for subexponential-hard semantical security).

Let us end this section by remarking that (sub-exponentially hard) semantical security trivially holds against polynomial-time “generic” attackers that are restricted to “legally” operating on the encodings—in fact, it holds even against *unbounded* generic attackers that are restricted to only making polynomially (or even subexponentially) many zero-test queries: recall that each legal zero-test query is constant with overwhelming probability (whether we operate on \vec{m}_0, \vec{z} or \vec{m}_1, \vec{z}) and thus by a Union Bound, the output of any generic attacker restricted to polynomially many zero-test queries is also constant with overwhelming probability; see Section 3.6 for a formal statement.

Semantical Security w.r.t. Restricted Classes of Message Samplers For our specific construction of indistinguishability obfuscators it suffices to assume the existence of *semantically secure encodings w.r.t. restricted classes of message samplers* M , where the $\{(\vec{S}_n, \vec{T}_n)\}_{n \in \mathbb{N}}$ -respecting condition on M is replaced by some stronger restriction on M . In particular, it suffices to restrict to message samplers M that induce a *high-entropy* distribution over $\vec{m}_0, \vec{m}_1, \vec{z}$ —not only the individual elements have high min-entropy but also any element computed by applying a “non-terminal” sequence of legal arithmetic operations to \vec{m}_b, \vec{z} (for $b \in \{0, 1\}$). More precisely, we say that a M is a *H-entropic* $\{(\vec{S}_n, \vec{T}_n)\}_{n \in \mathbb{N}}$ -respecting message

coding additionally release a random share of a secret-sharing of sp . If few shares are released (i.e., \vec{z} is small) security is untouched, but if many shares are released security is trivially broken.

sampler if M is $\{(\vec{S}_n, \vec{T}_n)\}_{n \in \mathbb{N}}$ -respecting, where the sets S_n and T_n are over the universe set $[k_n]$ and additionally:

- For every security parameter n , every $\text{pp} \in \text{InstGen}(1^n, 1^{k_n})$ describing a ring R , every non-terminal (\vec{S}_n, \vec{T}_n) -respecting arithmetic circuit C that computes a non-zero polynomial in its inputs, it holds that for $b \in \{0, 1\}$,

$$H_\infty(C(\vec{m}_b, \vec{z})) \geq H(\log |R|)$$

where $(\vec{m}_0, \vec{m}_1, \vec{z}) \leftarrow M(1^n, \text{pp})$.

We here focus on “very” high entropy message samplers, where $H(n) = n - O(\log n)$, and refer to such message samplers as simply *entropic* $\{(\vec{S}_n, \vec{T}_n)\}_{n \in \mathbb{N}}$ -respecting message sampler (or entropically valid), and refer to encoding schemes satisfying semantical security w.r.t. such restricted message samplers as *entropic semantically secure*.

Additionally, for our purposes, we may consider semantic security with respect to even more restricted types of message samplers M and sets (\vec{S}_n, \vec{T}_n) . In particular, where: (1) Each individual element sampled is statistically close to a uniform ring element; (2) Elements sampled are “almost” pair-wise independent: each pair of elements encoded is statistically close to two uniform ring elements;²³ (3) The sets contained in the sequences \vec{S}_n, \vec{T}_n are pairs of indices $\{i, j\}$, $i, j \in [k_n]$. Properties 1, 2 are natural abstractions of what happens in the GDDH assumption (property 2 is a relaxation of the independence, as opposed to just pair-wise independence, property satisfied by the GDDH assumption). Property 3 implies that (if we consider a arithmetic circuit) exactly $k/2$ multiplications on the elements must be performed before a zero-testing can be done;

²³We thank Hoeteck Wee to suggesting to consider independence properties among the elements.

combined with the above entropic message sampler condition, this implies that any set-respecting arithmetic circuit of multiplicative degree smaller than $k/2$ produces a high-entropy element when applied to the sampled elements.²⁴

3.4 iO from Semantically Secure Multilinear Encodings

In this section we prove that semantically secure multilinear encodings implies indistinguishability obfuscators for NC^1 . We will show this through the following steps.

- We first introduce a weaker notion of indistinguishability obfuscation for branching programs, which we call *neighboring-matrix indistinguishability obfuscation*. Roughly speaking, this notion guarantees that the obfuscations of any pair of functionally equivalent branching programs that *differ in only a few matrices* are computationally indistinguishable.
- We show that any neighboring-matrix indistinguishability obfuscator for branching programs can be transformed into an *full* indistinguishability obfuscator for NC^1 .
- Finally, we show that assuming the existence of semantically secure multilinear encodings, there exists a neighboring-matrix indistinguishability obfuscator for branching programs.

²⁴We thank Shai Halevi for this observation (and more generally for suggesting that we consider the output of low-degree arithmetic circuits as an alternative to our entropic condition.).

3.4.1 Neighboring-Matrix Indistinguishability Obfuscation (*nm-iO*)

We introduce a weaker notion of indistinguishability obfuscation for branching programs. This notion is similar to indistinguishability obfuscation except that instead of requiring security to hold with respect to *any* pair of functionally equivalent programs, we require security to hold with respect to any pair of *neighboring* programs that are functionally equivalent. We say a pair of branching programs are neighboring if they differ in only a few matrices.

Definition 36 (Neighboring-Matrix Branching Programs). *We say that BP_0 and BP_1 are a pair of neighboring-matrix branching programs if they differ in at most 4 matrices. We say that $\{BP_n^0\}_{n \in \mathcal{N}}$ and $\{BP_n^1\}_{n \in \mathcal{N}}$ are a pair of neighboring-matrix branching program ensembles if for every $n \in \mathcal{N}$, BP_n^0 and BP_n^1 are a pair of neighboring-matrix branching programs.*

Definition 37 (Neighboring-Matrix Indistinguishability Obfuscator). *A uniform PPT machine Obf is an neighboring-matrix indistinguishability obfuscator for an ensemble of classes of branching programs $\{\mathcal{B}_n\}_{n \in \mathbb{N}}$ if it satisfies the same correctness and security conditions as in Definition 24 except that the security condition quantifies only over pairs of neighboring-matrix branching program ensembles (as opposed to pairs of arbitrary circuit ensembles as in Definition 24).*

3.4.2 From *nm-iO* to *iO*

In this section we show that any neighboring-matrix indistinguishability obfuscator for a particular ensemble of classes of branching programs can be trans-

formed into *full* indistinguishability obfuscators for NC¹.

Roughly speaking, the indistinguishability obfuscator iO will use the neighboring-matrix indistinguishability obfuscator Obf in the following way: iO on input a circuit C , first converts it to an oblivious branching program BP using Theorem 20. Next, iO doubles the width of BP by “merging” it with a dummy branching program that computes the constant 1, and then adds a branch at the very start that chooses whether to use the true program or the dummy, based on a “switch”. iO simply returns the obfuscation of the above “merged” branching program as produced by Obf .

At a high level, to show indistinguishability of $iO(C_1)$ and $iO(C_2)$, our strategy will be to obfuscate (using Obf) the “merged” branching program for C_1 , and then, matrix by matrix, replace the dummy branching program with the branching program for C_2 . Once the entire dummy branching program has been replaced by C_2 , we flip the “switch” so that the composite branching program now computes the branching program for C_2 . We then replace the branching program for C_1 with C_2 , matrix by matrix, so that we have two copies of the branching program for C_2 . We now flip the “switch” again, and finally restore the dummy branching program, so that we end up with one copy of C_2 and one copy of the dummy. In this way, we transition from $iO(C_1)$ to $iO(C_2)$, while only changing a small piece of the branching program being obfuscated under Obf in each step, and keeping the functionality the same. If Obf is a neighboring-matrix indistinguishability obfuscator then each step of these transitions must be indistinguishable, hence showing iO is an *full* indistinguishability obfuscator.

Merging Branching Programs

We first describe a method Merge for combining any two matrix branching programs together to create a composite branching program of double width, in a way that enables switching by changing only a small number of matrices.

Construction 1 (Merging branching programs). *Let $BP_0 = \{\text{inp}(i), B_{i,0}^0, B_{i,1}^0\}_{i=1}^m$ and $BP_1 = \{\text{inp}(i), B_{i,0}^1, B_{i,1}^1\}_{i=1}^m$ be oblivious matrix branching programs, each of width w and length m for n input bits. (We assume that the same labelling function $\text{inp} : [m] \rightarrow [n]$ is used for each of BP_0 and BP_1 , and this is without loss of generality because we can add extra dummy levels so that this property holds.)*

Define branching programs $\widehat{BP}_0 = \{\text{inp}'(i), \hat{B}_{i,0}^0, \hat{B}_{i,1}^0\}_{i=1}^{m+2}$ and $\widehat{BP}_1 = \{\text{inp}'(i), \hat{B}_{i,0}^1, \hat{B}_{i,1}^1\}_{i=1}^{m+2}$, each of width $2w$ and length $m + 2$ on l input bits, where:

$$\text{inp}'(i) \stackrel{\text{def}}{=} \begin{cases} 1, & \text{when } i = 1 \\ \text{inp}(i - 1), & \text{when } 2 \leq i \leq m + 1 \\ 1, & \text{when } i = m + 2 \end{cases}$$

and, for all levels except the first and the last, \widehat{BP}_0 and \widehat{BP}_1 agree, given by:

$$\hat{B}_{i,b}^0 = \hat{B}_{i,b}^1 \stackrel{\text{def}}{=} \begin{pmatrix} B_{(i-1),b}^0 & 0 \\ 0 & B_{(i-1),b}^1 \end{pmatrix} \text{ for all } 2 \leq i \leq m + 1 \text{ and } b \in \{0, 1\}$$

and the first and last levels are given by:

$$\begin{aligned} \hat{B}_{1,b}^0 &= \hat{B}_{m+2,b}^0 = I_{2w \times 2w} && \text{for } b \in \{0, 1\} \\ \hat{B}_{1,b}^1 &= \hat{B}_{m+2,b}^1 = \begin{pmatrix} 0 & I_{w \times w} \\ I_{w \times w} & 0 \end{pmatrix} && \text{for } b \in \{0, 1\} \end{aligned}$$

We define Merge so that $\text{Merge}(BP_0, BP_1, 0) = \widehat{BP}_0$ and $\text{Merge}(BP_0, BP_1, 1) = \widehat{BP}_1$.

We will show that \widehat{BP}_0 and \widehat{BP}_1 are matrix branching programs that compute the same functions as BP_0 and BP_1 respectively, with the additional feature that \widehat{BP}_0 and \widehat{BP}_1 differ from each other in only two levels, namely the first and the last. Further, since inp' does not depend on the function being computed, \widehat{BP}_0 and \widehat{BP}_1 are *oblivious* matrix branching programs.

Accordingly, with respect to $\text{Merge}(BP_0, BP_1, b)$ we will often use the phrase *active branching program* to refer to BP_b .

Claim 22. For $BP_0 = \{\text{inp}(i), B_{i,0}^0, B_{i,1}^0\}_{i=1}^m$ and $BP_1 = \{\text{inp}(i), B_{i,0}^1, B_{i,1}^1\}_{i=1}^m$ each of width w and length m on n input bits, define \widehat{BP}_0 and \widehat{BP}_1 as above. Then, for each $b \in \{0, 1\}$, $x \in \{0, 1\}^n$,

$$\prod_{i=1}^{m+2} \widehat{B}_{i,x[\text{inp}'(i)]}^b = \begin{pmatrix} \prod_{i=1}^m B_{i,x[\text{inp}(i)]}^b & 0 \\ 10 & \prod_{i=1}^m B_{i,x[\text{inp}(i)]}^{1-b} \end{pmatrix}$$

Proof. We observe that \widehat{BP}_0 and \widehat{BP}_1 agree on each level except the first and last, that is,

$$\widehat{B}_{i,b}^0 = \widehat{B}_{i,b}^1 = \begin{pmatrix} B_{(i-1),b}^0 & 0 \\ 0 & B_{(i-1),b}^1 \end{pmatrix} \quad \forall \quad i : 2 \leq i \leq m+1, \quad b \in \{0, 1\}$$

Then we have, for any $x \in \{0, 1\}^n$,

$$\begin{aligned} \prod_{i=2}^{m+1} \widehat{B}_{i,x[\text{inp}'(i)]}^0 &= \prod_{i=2}^{m+1} \widehat{B}_{i,x[\text{inp}'(i)]}^1 = \prod_{i=2}^{m+1} \begin{pmatrix} B_{(i-1),x[\text{inp}'(i)]}^0 & 0 \\ 0 & B_{(i-1),x[\text{inp}'(i)]}^1 \end{pmatrix} \\ &= \prod_{i=1}^m \begin{pmatrix} B_{i,x[\text{inp}(i)]}^0 & 0 \\ 0 & B_{i,x[\text{inp}(i)]}^1 \end{pmatrix} \\ &= \begin{pmatrix} \prod_{i=1}^m B_{i,x[\text{inp}(i)]}^0 & 0 \\ 0 & \prod_{i=1}^m B_{i,x[\text{inp}(i)]}^1 \end{pmatrix} \end{aligned}$$

Where the change of indices in the second step follows because $\text{inp}'(i) = \text{inp}(i-1)$ when $2 \leq i \leq m+1$. We now consider the two case for $b \in \{0, 1\}$.

Case 1: (b = 0)

In this case,

$$\begin{aligned} \prod_{i=1}^{m+2} \widehat{B}_{i,x[\text{inp}'(i)]}^0 &= I_{2w \times 2w} \cdot \begin{pmatrix} \prod_{i=1}^m B_{i,x[\text{inp}(i)]}^0 & 0 \\ 0 & \prod_{i=1}^m B_{i,x[\text{inp}(i)]}^1 \end{pmatrix} \cdot I_{2w \times 2w} \\ &= \begin{pmatrix} \prod_{i=1}^m B_{i,x[\text{inp}(i)]}^0 & 0 \\ 0 & \prod_{i=1}^m B_{i,x[\text{inp}(i)]}^1 \end{pmatrix} \end{aligned}$$

as required.

Case 2: (b = 1)

In this case,

$$\begin{aligned} \prod_{i=1}^{m+2} \widehat{B}_{i,x[\text{inp}'(i)]}^1 &= \begin{pmatrix} 0 & I_{w \times w} \\ I_{w \times w} & 0 \end{pmatrix} \cdot \begin{pmatrix} \prod_{i=1}^m B_{i,x[\text{inp}(i)]}^0 & 0 \\ 0 & \prod_{i=1}^m B_{i,x[\text{inp}(i)]}^1 \end{pmatrix} \\ &= \begin{pmatrix} 0 & I_{w \times w} \\ I_{w \times w} & 0 \end{pmatrix} \\ &= \begin{pmatrix} 0 & \prod_{i=1}^m B_{i,x[\text{inp}(i)]}^1 \\ \prod_{i=1}^m B_{i,x[\text{inp}(i)]}^0 & 0 \end{pmatrix} \cdot \begin{pmatrix} 0 & I_{w \times w} \\ I_{w \times w} & 0 \end{pmatrix} \\ &= \begin{pmatrix} \prod_{i=1}^m B_{i,x[\text{inp}(i)]}^1 & 0 \\ 0 & \prod_{i=1}^m B_{i,x[\text{inp}(i)]}^0 \end{pmatrix} \end{aligned}$$

as required. □

Claim 23. For all BP_0 and BP_1 each of width w and length m on n input bits, for each $b \in \{0, 1\}$, for all $x \in \{0, 1\}^n$,

$$\text{Merge}(BP_0, BP_1, b)(x) = BP_b(x)$$

Proof. Let $BP_0 = \{\text{inp}(i), B_{i,0}^0, B_{i,1}^0\}_{i=1}^m$ and $BP_1 = \{\text{inp}(i), B_{i,0}^1, B_{i,1}^1\}_{i=1}^m$. Define $\widehat{BP}_0 = \text{Merge}(BP_0, BP_1, 0)$ and $\widehat{BP}_1 = \text{Merge}(BP_0, BP_1, 1)$ as above. We observe that for

any $x \in \{0, 1\}^n$,

$$\begin{aligned}
& \text{Merge}(BP_0, BP_1, b)(x) = 1 \\
& \iff \left(\prod_{i=1}^{m+2} \widehat{B}_{i,x[\text{inp}'(i)]}^b \right) \cdot \mathbf{e}_1 = \mathbf{e}_1 \\
& \iff \left(\begin{array}{cc} \prod_{i=1}^m B_{i,x[\text{inp}(i)]}^b & 0 \\ 0 & \prod_{i=1}^m B_{i,x[\text{inp}(i)]}^{1-b} \end{array} \right) \cdot \mathbf{e}_1 = \mathbf{e}_1 \quad (\text{from Claim 22}) \\
& \iff \left(\prod_{i=1}^m B_{i,x[\text{inp}(i)]}^b \right) \cdot \mathbf{e}_1 = \mathbf{e}_1 \\
& \iff BP_b(x) = 1
\end{aligned}$$

Thus $\text{Merge}(BP_0, BP_1, b)(x) = BP_b(x)$. \square

The following claim illustrates some useful properties of the Merge procedure that we will use later. Firstly it notes that changing the bit Merge gets as input changes only the “switch” matrices in the first and last level of the program Merge outputs. Secondly, changing one level of a program Merge gets as input changes the output program in one level only. Finally, the first column of the output matrix of the widened program output by Merge depends only on the first column of the output matrix of the active program. The claim follows by observing the definition of Merge.

Claim 24. *Let BP_0 and BP_1 be length m , width w branching programs, with input length n .*

- $\text{Merge}(BP_0, BP_1, 0)$ and $\text{Merge}(BP_0, BP_1, 1)$ differ in only 4 matrices, the matrices corresponding to the first and last level.

- Let BP'_1 be a length m branching program that differs from BP_1 in only the i^{th} level for some $i \in [m]$. Then for both $b \in \{0, 1\}$, $\text{Merge}(BP_0, BP_1, b)$ and $\text{Merge}(BP_0, BP'_1, b)$ also differ only in the i^{th} level. A similar statement holds for branching programs BP'_0 that differ from BP_0 in only one level.
- For any $b \in \{0, 1\}$, let $BP = \text{Merge}(BP_0, BP_1, b)$, and $P_{\text{out}}^{BP}(\cdot)$ and $P_{\text{out}}^{BP_b}(\cdot)$ be the functions computing the output matrices on a given input for BP and BP_b respectively. Then for every input $x \in \{0, 1\}^n$,

$$\text{col}_1(P_{\text{out}}^{BP}(x)) = \text{extend}(\text{col}_1(P_{\text{out}}^{BP_b}(x)))$$

where extend extends a length w vector by appending w zeroes to the end.

Let us emphasize that even if BP_0 and BP_1 have fixed accept and reject matrices, $\text{Merge}(BP_0, BP_1, b)$ may no longer be a branching program with fixed accept and reject matrices; however, it will be a branching program having fixed output column (as required by Definition 29).

The Construction

In this section we show how to construct an indistinguishability obfuscator for the class C^1 , given a neighboring-matrix indistinguishability obfuscator for branching programs Obf . By Lemma 19, $i\mathcal{O}$ can be converted into indistinguishability obfuscator for NC^1 .

Description of $i\mathcal{O}(1^n, C)$:

1. $i\mathcal{O}$ verifies that input $C \in C_n^1$ (that is, C is a circuit with size at most n and depth at most $\log(n)$), and aborts otherwise.

2. iO uses Barrington's Theorem to convert C into an oblivious width 5 permutation branching program. It pads this branching program as follows: First, it increases the number of input bits to the branching program to n . Next, it adds dummy levels to the end of the branching program until its length is the same as the longest branching program for a circuit in C_n^1 (which is $O(4^{\log(n)}) = O(n^2)$). Then, for every level in the branching program, it replaces it with n dummy levels that read every bit of the input in sequential order, inserting the original level into the corresponding position in this sequence.

This procedure ensures that every padded branching program for a circuit in C_n^1 has the same length, same number of input bits, and the same input labelling function inp as the padded branching program for any other circuit in C_n^1 . Let the padded branching program be $BP = \{\text{inp}(i), B_{i,0}, B_{i,1}\}_{i=1}^m$.

3. iO generates a dummy width-5 branching program $I = \{\text{inp}(i), I_{5 \times 5}, I_{5 \times 5}\}_{i=1}^m$ of length m , where each permutation matrix at each level is the identity matrix. iO then computes $\widehat{BP} = \text{Merge}(BP, I, 0)$.

4. iO outputs $\text{Obf}(\widehat{BP})$.

Proof of security

Theorem 25. *There exists a ensemble of classes of branching programs \mathcal{B} such that if there exists a neighboring-matrix indistinguishability obfuscator for \mathcal{B} then there exist indistinguishability obfuscators for NC^1 .*

Proof. We first define the ensemble of classes $\mathcal{B} = \{\mathcal{B}_n\}_{n \in \mathcal{N}}$. The class \mathcal{B}_n is simply the class of all matrix branching programs of width 10, and length $n^3 + 2$ such

that for every input x it holds that

$$\left(\prod_{i=1}^m B_{i,x[\text{inp}(i)]}\right) \cdot \mathbf{e}_1 = \begin{cases} \mathbf{e}_1 & \text{when } BP(x) = 1 \\ \mathbf{e}_k & \text{when } BP(x) = 0 \end{cases}$$

where $k \neq 1$ is such that $\mathbf{e}_k = \text{extend}(\mathbf{P}_{\text{reject}} \cdot \mathbf{e}_1)$ and $\mathbf{P}_{\text{reject}}$ is the rejecting matrix from Theorem 20.

Let Obf be a neighboring-matrix indistinguishability obfuscator for \mathcal{B} , and let $i\mathcal{O}$ be the obfuscator relying on Obf constructed in Section 3.4.2 . We will show $i\mathcal{O}$ is a indistinguishability obfuscator for C^1 ; by Lemma 19, this implies the existence of indistinguishability obfuscators for NC^1 .

Assume for contradiction that there exists a nuPPT distinguisher D and polynomial p such that for infinitely many n , there exist functionally equivalent circuits $C_n^0, C_n^1 \in C_n^1$ such that D distinguishes $i\mathcal{O}(1^n, C_n^0)$ and $i\mathcal{O}(1^n, C_n^1)$ with advantage $1/p(n)$. For any $n \in \mathbb{N}$, let BP_0 and BP_1 be the branching programs of length $m = \text{poly}(n)$ obtained by applying Theorem 20 to the circuits C_n^0 and C_n^1 respectively, and padding them so they have the same length and same input labelling function.

Let Hyb_i be a procedure that takes as input two length m branching programs P_0 and P_1 (with the same labeling function) and outputs a ‘‘hybrid’’ length m branching program whose first i levels are identical to the first i levels of P_0 and all the other levels are identical to those of P_1 . Formally, let $P_0 = \{\text{inp}(j), B_{j,0}, B_{j,1}\}_{j \in [m]}$ and $P_1 = \{\text{inp}(j), B'_{j,0}, B'_{j,1}\}_{j \in [m]}$.

$$\text{Hyb}_i(P_0, P_1) = \{\text{inp}(j), B_{j,0}, B_{j,1}\}_{j=1}^i, \{\text{inp}(j), B'_{j,0}, B'_{j,1}\}_{j=i+1}^m$$

For every $n \in \mathbb{N}$ we define hybrid distributions in the following way.

- We start with H_0 which is the obfuscation of the circuit C_n^0 .

$$H_0 = iO(1^n, C_n^0) = \text{Obf}(\text{Merge}(BP_0, I, 0))$$

- For $i = 1, 2 \dots m$, let

$$H_i = \text{Obf}(\text{Merge}(BP_0, \text{Hyb}_i(BP_1, I), 0))$$

We change, one level at a time, the second branching program Merge takes as input from I to BP_1 .

- We have that $H_m = \text{Obf}(\text{Merge}(BP_0, BP_1, 0))$. We change the “switch” input to Merge so that the second branching program BP_1 is active.

$$H_{m+1} = \text{Obf}(\text{Merge}(BP_0, BP_1, 1))$$

- For $i = 1, 2 \dots m$, let

$$H_{m+i+1} = \text{Obf}(\text{Merge}(\text{Hyb}_i(BP_1, BP_0), BP_1, 1))$$

We change the first program Merge takes as input from BP_0 to BP_1 , one level at a time as before.

- We have that $H_{2m+1} = \text{Obf}(\text{Merge}(BP_1, BP_1, 1))$. We switch back so that the first program is active (which in this case is the same as the second program BP_1)

$$H_{2m+2} = \text{Obf}(\text{Merge}(BP_1, BP_1, 0))$$

- For $i = 1, 2 \dots m$, let

$$H_{2m+i+2} = \text{Obf}(\text{Merge}(BP_1, \text{Hyb}_i(I, BP_1), 0))$$

We change the second program Merge takes as input from BP_1 to I , one level at a time as before. Finally we get

$$H_{3m+2} = iO(1^n, C_n^1) = \text{Obf}(\text{Merge}(BP_1, I, 0))$$

which is the obfuscation of the circuit C_n^1 .

Recall that by assumption D distinguishes between $\{iO(1^n, C_n^0)\}_{n \in \mathbb{N}}$ and $\{iO(1^n, C_n^1)\}_{n \in \mathbb{N}}$. That is, there is a polynomial p such that for infinitely many n

$$|\Pr[D(1^n, H_0) = 1] - \Pr[D(1^n, H_{3m+2}) = 1]| > 1/p(n)$$

By the above hybrid argument, D must distinguish between a pair of consecutive hybrids. That is, there exists some $i \in \{0, 1, \dots, 3m+1\}$ such that

$$|\Pr[D(1^n, H_i) = 1] - \Pr[D(1^n, H_{i+1}) = 1]| > 1/4mp(n)$$

We now show that H_i and H_{i+1} can be expressed as the $\text{Obf}(BP)$ and $\text{Obf}(BP')$ respectively where BP and BP' are relaxed matrix branching programs that differ in at most 4 matrices, agree on all inputs and come from \mathcal{B}_n .

Claim 26. *For every n , there exist branching programs $BP, BP' \in \mathcal{B}_n$ such that*

- $H_i = \text{Obf}(BP)$ and $H_{i+1} = \text{Obf}(BP')$.
- BP and BP' differ in at most 4 matrices.
- For all x , $BP(x) = BP'(x)$.

Proof. We consider three cases: when i is equal to m , $2m+1$ and otherwise.

Case 1: $i = m$: By definition of H_i and H_{i+1} , the branching programs BP and BP' are $\text{Merge}(BP_0, BP_1, 0)$ and $\text{Merge}(BP_0, BP_1, 1)$ respectively. By Claim 24, BP and BP' differ in the “switch” matrices, which make up 4 matrices (the first and last level). Furthermore, BP and BP' compute BP_0 and BP_1 respectively which are equivalent programs by assumption. It remains to show that $BP, BP' \in \mathcal{B}_n$. Note that BP and BP' have width 10 and length $n^3 + 2$. By Claim 24, the first column of the output matrix for a merged branching program only depends on the first

column of the output matrix of the active program. Hence, for every input x , $\text{col}_1(\mathbf{P}_{\text{out}}^{BP}(x)) = \text{extend}(\text{col}_1(\mathbf{P}_{\text{out}}^{BP_0}(x)))$. By Theorem 20, $\mathbf{P}_{\text{out}}^{BP_0}(x)$ is either $\mathbf{P}_{\text{accept}}$ or $\mathbf{P}_{\text{reject}}$ depending on the output $BP_0(x)$. Therefore, for all inputs x such that $BP(x) = 0$,

$$\text{col}_1(\mathbf{P}_{\text{out}}^{BP}(x)) = \text{extend}(\text{col}_1(\mathbf{P}_{\text{reject}})) = \mathbf{e}_k$$

Similarly, for all inputs x such that $BP(x) = 1$,

$$\text{col}_1(\mathbf{P}_{\text{out}}^{BP}(x)) = \text{extend}(\text{col}_1(\mathbf{P}_{\text{accept}})) = \mathbf{e}_1$$

The same argument holds for BP' too, in which case BP_1 is active and has the same accepting and rejecting permutations $\mathbf{P}_{\text{accept}}$ and $\mathbf{P}_{\text{reject}}$ by Theorem 20.

Case 2: $i = 2m + 1$: By definition of H_i and H_{i+1} , the branching programs BP and BP' are $\text{Merge}(BP_1, BP_1, 0)$ and $\text{Merge}(BP_1, BP_1, 1)$ respectively. As before, these programs differ in the 4 matrices only. Furthermore, both BP and BP' compute the same function, as the active program is the same (BP_1). Also as before, from Claim 24 and Theorem 20 we have that for all inputs x ,

$$\text{col}_1(\mathbf{P}_{\text{out}}^{BP}(x)) = \text{col}_1(\mathbf{P}_{\text{out}}^{BP'}(x)) = \text{extend}(\text{col}_1(\mathbf{P}_{\text{out}}^{BP_1}(x))) = \mathbf{e}_t$$

where $t = 1$ if $BP_1(x) = 1$ and $t = k$ otherwise.

Case 3: $i \neq m$ and $i \neq 2m + 1$: First, consider the subcase when $i < m$ or $i > 2m + 1$. The programs BP and BP' are of the form $\text{Merge}(BP_0, P_i)$ and $\text{Merge}(BP_0, P_{i+1})$ respectively where P_i and P_{i+1} are branching programs that differ only in the $i + 1^{\text{th}}$ level. By Claim 24, BP and BP' differ only in the $i + 1^{\text{th}}$ level too. Furthermore, in both BP and BP' , the active program is BP_0 . Hence BP and BP' compute the same function and similarly as the previous case, we have that for all inputs x ,

$$\text{col}_1(\mathbf{P}_{\text{out}}^{BP}(x)) = \text{col}_1(\mathbf{P}_{\text{out}}^{BP'}(x)) = \text{extend}(\text{col}_1(\mathbf{P}_{\text{out}}^{BP_0}(x))) = \mathbf{e}_t$$

where $t = 1$ if $BP_1(x) = 1$ and $t = k$ otherwise. The case when $m < i < 2m + 1$ follows similarly. This concludes the proof of the claim. \square

Therefore we have that there is a polynomial p' such that for infinitely many n there exist functionally equivalent branching programs $BP, BP' \in \mathcal{B}_n$ that differ in only a few matrices such that

$$|Pr[D(1^n, \text{Obf}(BP)) = 1] - Pr[D(1^n, \text{Obf}(BP'))]| > 1/p'(n)$$

This implies Obf is not a neighboring-matrix indistinguishability obfuscator for \mathcal{B} and hence a contradiction. \square

3.4.3 From Semantic Security to $nm-iO$

In this section we show that assuming the existence of semantically secure multilinear encodings, there exists a neighboring-matrix indistinguishability obfuscator for any ensemble of classes of branching programs.

As in previous works [57, 36, 9], the strategy for our construction will be to apply Kilian's randomization technique to the matrices, and then encode these matrices using the graded encoding scheme. The encoding will be using a so-called "straddling set system" (as in [9]) that will enforce that any arithmetic circuit operating on these encodings can be decomposed into a sum of terms such that each term can be expressed using only encodings that come from one branch of the branching program (more specifically, from the path through the branching program corresponding to evaluating a particular input x to the branching program).

As mentioned in the introduction, although we will closely follow techniques from [36, 9] (our obfuscator may be viewed as a simplified version of the obfuscator from [9]), we cannot directly rely on their proofs for two reasons:

1. The proofs in [36, 9] rely on the fact that we are only obfuscating branching programs with fixed accept and reject matrices; as mentioned, we need to handle more general classes of branching programs.
2. The proofs in [36, 9] only reason about *polynomial-size* generic attackers. In contrast, to rely on semantical security, we need to reason about *unbounded* arithmetic circuits.

Randomizing Branching Programs

We start by describing Kilian’s randomization technique [86] for a branching program, adapted to our setting, by defining a process Rand that randomizes the matrices of a branching program BP . We will decompose the randomization into two parts, Rand^B and Rand^α , defined below, and define Rand as their composition.

Definition 38 (Rand^B). *Let $BP = \{\text{inp}(i), B_{i,0}, B_{i,1}\}_{i=1}^m$ be a branching program of width w and length m , with length- n inputs. Let p be a prime exponential in n . Then the process $\text{Rand}^B(BP, p)$ samples m random invertible matrices $R_1, R_2, \dots, R_m \in Z_p^{w \times w}$ uniformly and independently, and computes*

$$\tilde{B}_{i,b} = R_{(i-1)} \cdot B_{i,b} \cdot R_i^{-1} \quad \text{for every } i \in [m], \text{ and } b \in \{0, 1\}$$

where R_0 is defined as $I_{w \times w}$, and

$$\mathbf{t} = R_m \cdot \mathbf{e}_1$$

Rand^B then outputs

$$(\{\tilde{B}_{i,b}\}_{i \in [m], b \in \{0,1\}}, \mathbf{t}, p)$$

Definition 39 (Rand^α). Let $(\{\tilde{B}_{i,b}\}_{i \in [m], b \in \{0,1\}}, \mathbf{t}, p)$ be the output of $\text{Rand}^B(BP, p)$ as defined above. On this input, $\text{Rand}^\alpha(\{\tilde{B}_{i,b}\}_{i \in [m], b \in \{0,1\}}, p)$ samples $2m$ non-zero scalars $\{\alpha_{i,b} \in \mathcal{Z}_p : i \in [m], b \in \{0,1\}\}$ uniformly and independently, and outputs

$$(\{\alpha_{i,b} \cdot \tilde{B}_{i,b}\}_{i \in [m], b \in \{0,1\}}, \mathbf{t})$$

Definition 40 (Rand). Let $BP = \{\text{inp}(i), B_{i,0}, B_{i,1}\}_{i=1}^m$ be a branching program of width w and length m , with length- n inputs. Let p be a prime exponential in n . Then we define $\text{Rand}(BP, p)$ to be:

$$\begin{aligned} \text{Rand}(BP, p) &= (\text{Rand}^\alpha(\text{Rand}^B(BP, p))) \\ &= (\{\alpha_{i,b} \cdot \tilde{B}_{i,b}\}_{i \in [m], b \in \{0,1\}}, \mathbf{t}) \end{aligned}$$

Where $(\{\alpha_{i,b} \cdot \tilde{B}_{i,b}\}_{i \in [m], b \in \{0,1\}}, \mathbf{t})$ are as computed in the executions of Rand^α and Rand^B .

Execution of a randomized branching program: To compute $BP(x)$ from the output of $\text{Rand}(BP, p)$, given some input labelling function $\text{inp} : [m] \rightarrow [n]$, and $x \in \{0,1\}^n$, we compute

$$\text{Out}(x) = \left(\prod_{i=1}^m \alpha_{i,x[\text{inp}(i)]} \cdot \tilde{B}_{i,x[\text{inp}(i)]} \right) \cdot \mathbf{t}$$

Where $\text{Out} \in \mathbb{Z}_p^w$ is a $w \times 1$ vector. The intermediate multiplications cause each R_i^{-1} to cancel each R_i , and $R_0 = I_{w \times w}$, so the above computation can also be expressed as:

$$\text{Out}(x) = \left(\prod_{i=1}^m \alpha_{i,x[\text{inp}(i)]} \cdot B_{i,x[\text{inp}(i)]} \right) \cdot \mathbf{e}_1$$

When $BP(x) = 1$, we have that

$$\prod_{i=1}^m \alpha_{i,x[\text{inp}(i)]} \cdot B_{i,x[\text{inp}(i)]} \cdot \mathbf{e}_1 = \left(\prod_{i=1}^m \alpha_{i,x[\text{inp}(i)]} \right) \cdot \mathbf{e}_1$$

When $BP(x) = 0$, we have that

$$\prod_{i=1}^m \alpha_{i,x[\text{inp}(i)]} \cdot B_{i,x[\text{inp}(i)]} \cdot \mathbf{e}_1 = \left(\prod_{i=1}^m \alpha_{i,x[\text{inp}(i)]} \right) \cdot \mathbf{e}_k$$

for $k \neq 1$. Hence, to compute $BP(x)$, we compute $\text{Out}(x)$ and output 0 if $\text{Out}(x)[1] = 0$, and 1 otherwise.

Simulating a randomized branching program: Previous works ([9, 36]) followed [86] to show how to simulate the distribution of any single path corresponding to an input x using just $BP(x)$. However, the simulator required that branching programs have unique accept and reject matrices $\mathbf{P}_{\text{accept}}$ and $\mathbf{P}_{\text{reject}}$.

We would also like a theorem, along the lines of [86], that shows that any single path through a randomized branching program BP corresponding to an input x can be simulated knowing just the accept/reject behavior of BP on x (i.e. by knowing whether $BP(x) = 1$).

In our setting, however, branching programs only meet the relaxed requirement that the output matrix $\mathbf{P}_{\text{out}(x)}$ computed by evaluating BP on input x satisfies $\mathbf{P}_{\text{out}(x)} \cdot \mathbf{e}_1 = \mathbf{e}_1 \iff BP(x) = 1$. There can thus be multiple accept and reject matrices, and the particular accept or reject matrix output by BP can depend both on x and on the specific implementation of BP (and not simply its accept/reject behavior). In contrast, in previous works, because $\mathbf{P}_{\text{accept}}$ and $\mathbf{P}_{\text{reject}}$ were unique, knowing just the accept/reject behavior of BP on x also determines $\mathbf{P}_{\text{out}(x)}$.

What we will show is that, for the particular randomization scheme chosen above, we can simulate any single path through a randomized branching program BP corresponding to an input x without knowing the exact accept/reject

matrix $P_{\text{out}(x)}$, but rather just knowing the first column $\mathbf{p}_{\text{out}(x)} = \text{col}_1(P_{\text{out}(x)})$.

This will be sufficient for our applications, because the class of branching programs we randomize will have the property that there are fixed columns $\mathbf{p}_{\text{accept}}$ and $\mathbf{p}_{\text{reject}} \in \mathbb{Z}_p^w$ such that for all $x \in \{0, 1\}^n$, if $BP(x) = 1$ then $\text{col}_1(P_{\text{out}(x)}) = \mathbf{p}_{\text{accept}}$, and if $BP(x) = 0$ then $\text{col}_1(P_{\text{out}(x)}) = \mathbf{p}_{\text{reject}}$. In the case of such programs, $\text{col}_1(P_{\text{out}(x)})$ is determined solely by $BP(x)$, and not the particular implementation of BP . Thus, for these programs, we can simulate given only $BP(x)$.

Before we show this theorem, we define notation for a path through a branching program corresponding to an input x .

Definition 41 (proj_x). *Let $\text{inp} : [m] \rightarrow [n]$ be an input labelling function, and, for any $x \in \{0, 1\}^n$, define proj_x , relative to inp , such that for any branching program BP with labelling function inp , for any prime $p \in \mathcal{N}$, and for any $(\{\tilde{B}_{i,b}\}_{i \in [m], b \in \{0,1\}}, \mathbf{t}) \leftarrow \text{Rand}^B(BP, p)$*

$$\text{proj}_x(\{\tilde{B}_{i,b}\}_{i \in [m], b \in \{0,1\}}, \mathbf{t}) = (\{\tilde{B}_{i,x[\text{inp}(i)]}\}_{i \in [m]}, \mathbf{t}),$$

that is, proj_x selects the elements from $(\{\tilde{B}_{i,b}\}_{i \in [m], b \in \{0,1\}}, \mathbf{t})$ used when evaluating input x .

We now show a version of Kilian's theorem, adapted to our construction.

Theorem 27. *There exists an efficient simulator KSim such that the following holds. Let $BP = \{\text{inp}(i), B_{i,0}, B_{i,1}\}_{i \in [m]}$ be a width- w branching program of length m on n bit inputs, and p a prime exponential in n . Let $x \in \{0, 1\}^n$ be an input to BP , and let $b_i = x[\text{inp}(i)]$ for each $i \in [m]$. Let $P_{\text{out}(x)} = \prod_{i=1}^m B_{i,b_i}$ denote the matrix obtained by evaluating BP on x , and let $\mathbf{p}_{\text{out}(x)} = \text{col}_1(P_{\text{out}(x)})$ denote the first column of this output. Let $\text{proj}_x(\text{Rand}^B(BP, p))$ be defined respecting the labelling function inp . Then $\text{KSim}(1^m, p, \mathbf{p}_{\text{out}(x)})$ is identically distributed to $\text{proj}_x(\text{Rand}^B(BP, p))$.*

Proof. We begin by defining $\text{KSim}(1^n, p, BP(x))$ as follows:

- For each i , KSim selects \tilde{B}_{i,b_i} to be a uniformly random invertible matrix in $\mathcal{Z}_p^{w \times w}$.
- KSim selects $\mathbf{t} \in \mathcal{Z}_p^w$ solving

$$\left(\prod_{i \in [m]} \tilde{B}_{i,b_i} \right) \cdot \mathbf{t} = \mathbf{p}_{\text{out}}(x) \quad (3.1)$$

where $b_i = x[\text{inp}(i)]$ for each i .

- KSim outputs $\{\{\tilde{B}_{i,b_i}\}_{i \in [m]}, \mathbf{t}\}$

We want to show that the distribution output by KSim matches the real distribution of $\{\{\tilde{B}_{i,b_i}\}_{i \in [m]}, \mathbf{t}\}$ in the output of $\text{Rand}^B(BP, p)$. But from [86], we have the following:

Claim 28. *The distribution of $\{\{\tilde{B}_{i,b_i}\}_{i \in [m]}, R_m\}$ can be exactly sampled given $\mathbf{P}_{\text{out}}(x)$, by sampling $\{\tilde{B}_{i,b_i}\}_{i \in [m]}, R_m$ to be uniformly random and independent invertible matrices in $\mathcal{Z}_p^{w \times w}$ subject to*

$$\left(\prod_{i \in [m]} \tilde{B}_{i,b_i} \right) \cdot R_m = \mathbf{P}_{\text{out}}(x) \quad (3.2)$$

The above claim implies the following:

Claim 29. *The distribution of $\{\{\tilde{B}_{i,b_i}\}_{i \in [m]}, R_m\}$ can be sampled by independently choosing each \tilde{B}_{i,b_i} uniform and invertible, and fixing R_m solving equation (3.2).*

Proof. This follows because for every choice of invertible \tilde{B}_{i,b_i} , there exists R_m solving equation (3.2) given by

$$R_m = \left(\prod_{i \in [m]} \tilde{B}_{i,b_i} \right)^{-1} \cdot \mathbf{P}_{\text{out}}(x) \quad (3.3)$$

Further, every solution to equation (3.2) can be represented as invertible \tilde{B}_{i,b_i} , and an R_m solving equation (3.3). Thus choosing a random solution to equation (3.2) corresponds to independently choosing each \tilde{B}_{i,b_i} uniformly and invertible, and fixing R_m solving equation (3.3). \square

From the above argument, we have that the distribution of $\text{proj}_x(\text{Rand}(BP, p))$ is exactly the same as the distribution produced by independently choosing each \tilde{B}_{i,b_i} uniform and invertible, fixing R_m solving equation (3.3), setting \mathbf{t} to be the first column of R_m , and outputting $\{ \{ \tilde{B}_{i,b_i} \}_{i \in [m]}, \mathbf{t} \}$. But note that each column $\text{col}_i(R_m), i \in [w]$ is the unique solution to

$$\left(\prod_{i \in [m]} \tilde{B}_{i,b_i} \right) \cdot \text{col}_i(R_m) = \text{col}_i(\mathbf{P}_{\text{out}}(x))$$

Thus we have that each \tilde{B}_{i,b_i} is independent, uniform, and invertible, and, using $i = 1$, \mathbf{t} is the unique solution to

$$\left(\prod_{i \in [m]} \tilde{B}_{i,b_i} \right) \cdot \mathbf{t} = \mathbf{p}_{\text{out}}(x)$$

and, in particular, that \mathbf{t} is determined by *only* the first column of $\mathbf{P}_{\text{out}}(x)$. Thus, we see that the distribution of $\text{proj}_x(\text{Rand}^B(BP, p))$ is exactly the same as that output by KSim. \square

Choosing a Set System

In this section we will describe how to choose a collection of sets under which to encode a randomized branching program using the graded encoding scheme. Our selection of sets will closely follow [9], in that we use straddling set systems. However, one difference is that while they use dual input branching programs,

we restrict our attention to single-input schemes. As a consequence, the sets will be simpler and consist of fewer elements.

We first define straddling set systems.

Definition 42 (Straddling Set Systems [9]). *A straddling set system with n entries is a collection of sets $\mathbb{S}_n = \{S_{i,b} : i \in [n], b \in \{0, 1\}\}$ over a universe U , such that:*

$$\bigcup_{i \in [n]} S_{i,0} = \bigcup_{i \in [n]} S_{i,1} = U$$

and for every distinct non-empty sets $C, D \subseteq \mathbb{S}_n$, we have that if:

1. (Disjoint Sets:) C contains only disjoint sets. D contains only disjoint sets.
2. (Collision:) $\bigcup_{S \in C} S = \bigcup_{S \in D} S$

Then it must be that $\exists b \in \{0, 1\}$ such that:

$$C = \{S_{j,b}\}_{j \in [n]} \quad , \quad D = \{S_{j,(1-b)}\}_{j \in [n]}$$

Informally, the guarantee provided by a straddling set system is that only way to exactly cover U using elements from \mathbb{S}_n is to use either all sets $\{S_{i,0}\}_{i \in [n]}$ or all sets $\{S_{i,1}\}_{i \in [n]}$. We use a slight variant of their construction, choosing U to be $[2n]$, each $S_{i,0}$ to be one of $\{1, 2\}, \{3, 4\}, \dots, \{2n-1, 2n\}$, and each $S_{i,1}$ to be one of $\{1, 2n\}, \{2, 3\}, \{4, 5\} \dots, \{2n-2, 2n-1\}$.²⁵ By a proof exactly following [9], we have that this construction is a straddling set system.

Theorem 30 (Following Construction 1 in [9]). *For every $n \in \mathbb{N}$, there exists a straddling set system \mathbb{S}_n with n entries, over a universe U of $2n$ elements; furthermore, each set in the straddling set system has size exactly two.*

²⁵In the construction of [9], $U = [2n-1]$, and each $S_{i,0}$ is one of $\{1\}, \{2, 3\}, \dots, \{2n-2, 2n-1\}$, and each $S_{i,1}$ is one of $\{1, 2\}, \{3, 4\}, \dots, \{2n-1\}$. We could have also worked with this construction, but modify it slightly to ensure that all encodings are under sets of size exactly two.

We now define the process `SetSystem` which takes as input the length m of a branching program, the number of input bits n , and the input labelling function $\text{inp} : [m] \rightarrow [n]$ for a branching program. `SetSystem` will output the collection of straddling set systems that we will use to encode any branching program of length m on n input bits, with labelling function inp .

Execution of `SetSystem`(m, n, inp):

We let n_j denote the number of levels that inspect the j th input bit in inp . That is,

$$n_j = |\{i \in [m] : \text{inp}(i) = j\}|$$

For every $j \in [n]$, `SetSystem` chooses \mathbb{S}^j to be a straddling set system with n_j entries over a set U_j , such that the sets U_1, \dots, U_n are disjoint. Let $U = \bigcup_{j \in [n]} U_j$. `SetSystem` then chooses S_t be a set of two elements²⁶, disjoint from U . We associate the set system \mathbb{S}^j with the j 'th input bit of the branching program corresponding to inp . `SetSystem` then re-indexes the elements of \mathbb{S}^j to match the steps of the branching program as described by inp , so that:

$$\mathbb{S}^j = \{S_{i,b} : \text{inp}(i) = j, b \in \{0, 1\}\}$$

By this indexing, we also have that $S_{i,b} \in \mathbb{S}^{\text{inp}(i)}$ for every $i \in [m]$, for every $b \in \{0, 1\}$.

Let $k = |U \cup S_t|$, and WLOG, assume that the U^j s and S_t are disjoint subsets of $[k]$ (otherwise `SetSystem` relabels the elements to satisfy this property).

`SetSystem` then outputs

$$k, \quad \{S_{i,b}\}_{i \in [m], b \in \{0,1\}}, \quad S_t$$

²⁶We make this choice to ensure that every set in the output of `SetSystem` consists of exactly two indices $\{i, j\}$ for $i, j \in [k]$

The Construction

We finally describe our neighboring-matrix indistinguishability obfuscator Obf for branching programs. Obf will use Rand and SetSystem as subroutines.

Description of $\text{Obf}(BP)$:

Input. Obf takes as input an oblivious permutation branching program $BP = \{\text{inp}(i), B_{i,0}, B_{i,1}\}_{i=1}^m$ of width w , length m and taking n input bits.

Choosing sets. Obf runs $\text{SetSystem}(m, n, \text{inp})$ and receives $k, \{S_{i,b}\}_{i \in [m+2], b \in \{0,1\}}, S_t$.

Initializing the GES. Obf runs $\text{InstGen}(1^n, 1^k)$ and receives secret parameters sp and public parameters pp which describe a (k, R) -graded encoding scheme.

We assume the ring R is equal to \mathbb{Z}_p for some p exponential in n and k .

Randomizing BP. Obf executes $\text{Rand}(BP, p)$, and obtains its output, $\{\{\text{inp}(i), \alpha_{i,0} \cdot \tilde{B}_{i,0}, \alpha_{i,1} \cdot \tilde{B}_{i,1}\}_{i \in [m]}, \mathbf{t}\}$

Output. Obf outputs:

$$\text{pp}, \quad \{\text{inp}(i), \quad \text{Enc}(\text{sp}, \alpha_{i,0} \cdot \tilde{B}_{i,0}, S_{i,0}), \quad \text{Enc}(\text{sp}, \alpha_{i,0} \cdot \tilde{B}_{i,0}, S_{i,1})\}_{i \in [m]}, \quad \text{Enc}(\text{sp}, \mathbf{t}, S_t)$$

We also define a generic version of Obf , which we refer to as GObf . Its output will be used to initialize an oracle \mathcal{M} for the idealized version of the graded encoded scheme. $\text{GObf}(BP, \text{pp})$ acts exactly as $\text{Obf}(BP)$, except that it works with a fixed public parameter pp supplied as input, and in the **Output** step, GObf outputs

$$\text{pp}, \quad \{\text{inp}(i), (\alpha_{i,0} \cdot \tilde{B}_{i,0}, S_{i,0}), (\alpha_{i,1} \cdot \tilde{B}_{i,1}, S_{i,1})\}_{i \in [m]}, \quad (\mathbf{t}, S_t)$$

that is, the output before it is encoded under the multilinear encoding scheme.

Proof of security

We show that Obf defined in Section 3.4.3 is a neighboring-matrix indistinguishability obfuscator for any ensemble of classes of branching programs, if the underlying multilinear encodings are semantically secure.

Theorem 31. *Assume the existence of an entropic semantically secure multilinear encoding scheme. Then there exist a neighboring-matrix indistinguishability obfuscator for any ensemble of classes of branching programs.*

Proof. Consider any ensemble $\mathcal{B} = \{\mathcal{B}_n\}_{n \in \mathbb{N}}$ of classes of branching programs. We show that the obfuscator Obf is a neighboring-matrix indistinguishability obfuscator for \mathcal{B} . Assume for contradiction there exist a pair of ensembles $\{BP_n^0\}_{n \in \mathbb{N}}, \{BP_n^1\}_{n \in \mathbb{N}}$ nuPPT D and polynomial p such that for infinitely many n , BP_n^0, BP_n^1 are functionally equivalent programs in \mathcal{B}_n that differ in at most 4 matrices and

$$|\Pr[D(1^n, \text{Obf}(BP_n^0)) = 1] - \Pr[D(1^n, \text{Obf}(BP_n^1)) = 1]| > 1/p(n)$$

We will show that the semantic security of the multilinear encodings used by Obf implies a contradiction. In particular, we construct a message sampler M which samples $(\vec{m}_0, \vec{m}_1, \vec{z})$ such that $\text{Obf}(BP_n^0)$ is simply the encoding of (\vec{m}_0, \vec{z}) and $\text{Obf}(BP_n^1)$ is the encoding of (\vec{m}_1, \vec{z}) . We then show that if BP_n^0 and BP_n^1 agree on all inputs, then the message sampler M is valid in the sense of Definition 34 and therefore D breaks the semantic security of the encoding scheme used, hence a contradiction.

Fix $n \in \mathbb{N}$, and let $BP_n^0 = \{\text{inp}(i), B_{i,0}, B_{i,1}\}_{i \in [m]}$ and $BP_n^1 = \{\text{inp}(i), B'_{i,0}, B'_{i,1}\}_{i \in [m]}$. Let $L \subset [m] \times \{0, 1\}$ be the set of indices of those matrices in which BP_n^0 and BP_n^1 differ.

Note that by assumption $|L| = 4$. All other matrices of BP_n^0 and BP_n^1 are the same.

Let $(k, \{S_{i,b}\}_{i \in [m], b \in \{0,1\}}, S_{\mathbf{t}}) = \text{SetSystem}(m, n', \text{inp})$ where n' is the input length of the branching programs BP_n^0, BP_n^1 , and let

$$\vec{S}_n = \{S_l\}_{l \in L}$$

$$\vec{T}_n = (\{S_l\}_{l \notin L}, S_{\mathbf{t}})$$

We now define a message sampler M as follows. When run with security parameter 1^n , M gets BP_n^0 and BP_n^1 as non-uniform advice. On input 1^n , public parameters pp that describe a (k, \mathbb{Z}_p) -graded encoding scheme, M samples m random invertible 10×10 matrices over \mathbb{Z}_p , $\{R_i\}_{i \in [m]}$ and $2m$ random scalars from \mathbb{Z}_p , $\{\alpha_{i,b}\}_{i \in [m], b \in \{0,1\}}$. M then uses these matrices and scalars to randomize BP_n^0 and BP_n^1 as described by $\text{Rand}(\cdot, p)$ to obtain $\{\alpha_{i,b} \cdot \tilde{B}_{i,b}\}_{i \in [m'], b \in \{0,1\}}$, $\{\alpha_{i,b} \cdot \tilde{B}'_{i,b}\}_{i \in [m'], b \in \{0,1\}}$ and \mathbf{t} . M outputs

$$\vec{m}_0 = \{\alpha_l \cdot \tilde{B}_l\}_{l \in L}$$

$$\vec{m}_1 = \{\alpha_l \cdot \tilde{B}'_l\}_{l \in L}$$

$$\vec{z} = (\{\alpha_l \cdot \tilde{B}_l\}_{l \notin L}, \mathbf{t})$$

We observe that $D(1^n, \text{Obf}(BP_n^b))$ is simply the output of D when playing the semantic security game in Definition 35 parameterized by the bit b with the message sampler M and sets (\vec{S}_n, \vec{T}_n) (as defined above). To see this, observe that the distribution of (\vec{m}_0, \vec{z}) is identical to $\text{Rand}(BP_n^0, p)$ and the distribution of (\vec{m}_1, \vec{z}) is identical to $\text{Rand}(BP_n^1, p)$. When these elements are encoded under sets \vec{S}_n, \vec{T}_n then we obtain the distributions $\text{Obf}(BP_n^0)$ and $\text{Obf}(BP_n^1)$ respectively.

Recall that for infinitely many n ,

$$|\text{Pr}[D(1^n, \text{Obf}(BP_n^0)) = 1] - \text{Pr}[D(1^n, \text{Obf}(BP_n^1)) = 1]| > 1/p(n)$$

Since the graded encoding scheme is semantically secure, and $|\vec{S}_n| \in O(1)$ and $|\vec{T}_n| \in O(k)$, it must be that M is not a $\{\vec{S}_n, \vec{T}_n\}_{n \in \mathbb{N}}$ -respecting message sampler. In the remainder of the proof we show that if BP and BP' agree on all inputs then M is a $\{\vec{S}_n, \vec{T}_n\}_{n \in \mathbb{N}}$ -respecting message sampler, hence implying a contradiction. Similar statements were shown in [9] and [36]. In particular, GObf is a simplified version of the obfuscator of [9], which [9] shows is VBB secure against algebraic adversaries. We will follow the structure of the proof in [9], but cannot use it in a black-box way due to the differences in the construction and the fact that their proof only works for branching programs that have unique accepting and rejecting output matrices. The branching programs we consider may not have this property.

To prove that M is a $\{\vec{S}_n, \vec{T}_n\}_{n \in \mathbb{N}}$ -respecting message sampler we need to show that there exists a polynomial Q such that for every $n \in \mathbb{N}$, every (sp, pp) in the support of $\text{InstGen}(1^n, 1^k)$, and every (\vec{S}_n, \vec{T}_n) -respecting arithmetic circuit C , there exists a constant $c \in \{0, 1\}$ such that for any $b \in \{0, 1\}$,

$$\Pr[(\vec{m}_0, \vec{m}_1, \vec{z}) \leftarrow M(1^n, \text{pp}) : \text{isZero}(C(\vec{m}_b, \vec{z})) = c] \geq 1 - Q(n, k)/|R|.$$

where R is the ring associated with pp . We show that the result of applying any (\vec{S}_n, \vec{T}_n) -respecting arithmetic circuit C on (\vec{m}_0, \vec{z}) (resp. (\vec{m}_1, \vec{z})), can be *simulated* with overwhelming probability given just BP_n^0 . This implies (by a union bound over $b \in \{0, 1\}$) that for every such C there exists some bit c such that with overwhelming probability $C(\vec{m}_b, \vec{z}) = c$ for $b \in \{0, 1\}$, and thus M is $\{\vec{S}_n, \vec{T}_n\}_{n \in \mathbb{N}}$ -respecting. It suffices to show the following lemma and to note that BP_n^0 and BP_n^1 are functionally equivalent.

Lemma 32. *There exists a Turing machine CSim such that for every $m, n, w \in \mathbb{N}$, $v_0, v_1 \in \{0, 1\}^w$, labeling function $\text{inp} : [m] \rightarrow [n]$, prime number p , and \vec{S} -respecting*

arithmetic circuit C where $\vec{S} = \text{SetSystem}(m, n, \text{inp})$, the following holds. For every branching program BP of length m , width w and labeling function inp for which on every input x , $\text{col}_1(\text{P}_{\text{out}}(x)) = v_{BP(x)}$ it holds that

$$\Pr[\text{isZero}(C(\text{Rand}(BP, p))) \neq \text{CSim}^{BP}(1^m, p, C, v_0, v_1)] \leq 32wm/p$$

The proof of the lemma follows the structure of the VBB simulation in [9], appropriately adapted to deal with the fact that our branching programs do not have a unique output by relying on Theorem 27.

Proof. Roughly speaking the lemma follows from the the property that \vec{S} -respecting arithmetic circuits, due to the straddling set systems in \vec{S} , can only evaluate expressions that are “consistent” with some inputs. In particular, following [9], the polynomial evaluated by C can be expressed as the sum of *single-input terms* where each *single-input term* is a function of elements that are consistent with some single input to the branching program. Next, we rely on Theorem 27 to show that the sum of these single-input terms will depend only on the value of the branching program on these inputs.

The following proposition states that the function a \vec{S} -respecting arithmetic circuit computes can be expressed as the sum of several *single-input terms*. This decomposition is very similar to the one shown in [9].²⁷

Proposition 1. Fix $m, n, w \in \mathcal{N}$ and $\text{inp} : [m] \rightarrow [n]$. Let $\vec{S} = \text{SetSystem}(m, n, \text{inp}) = (\{S_{i,b}\}_{i \in [m], b \in \{0,1\}}, S_t)$, and let C be any \vec{S} -respecting arithmetic circuit. There exists a set $X \subseteq \{0, 1\}^n$ of inputs such that

²⁷The key difference is that [9] proves such a decomposition for “dual-input” branching program, and use the “dual-input” property to show that there are only polynomially many terms in the decomposition.

(i)

$$C(\{\alpha_{i,b} \tilde{B}_{i,b}\}_{i \in [m], b \in \{0,1\}}, \mathbf{t}) \equiv \sum_{x \in X} C_x(\{\alpha_{i,b} \tilde{B}_{i,b}\}_{i \in [m], b \in \{0,1\}}, \mathbf{t})$$

where each C_x is a \vec{S} -respecting arithmetic circuit, whose input wires are labelled only with sets respecting a single input $x \in \{0, 1\}^n$, that is, only with sets $\in \{S_{i,x[\text{inp}(i)]}\}_{i \in [m]} \cup \{S_t\}$.

(ii) For each C_x above, for every branching program BP of width w and length m on n input bits, with input labelling function inp , every prime p , and every $(\{\alpha_{i,b} \cdot \tilde{B}_{i,b}\}_{i \in [m], b \in \{0,1\}}, \mathbf{t}) \leftarrow \text{Rand}(BP, p)$

$$C_x(\{\alpha_{i,b} \tilde{B}_{i,b}\}_{i \in [m], b \in \{0,1\}}, \mathbf{t}) = \alpha_x \cdot p_x(\{\tilde{B}_{i,x[\text{inp}(i)]}\}_{i \in [m]}, \mathbf{t})$$

where p_x is some polynomial, and $\alpha_x = (\prod_{i \in [m]} \alpha_{i,x[\text{inp}(i)]})$. Furthermore, when p_x is viewed as a sum of monomials, each monomial contains exactly one entry from each $\tilde{B}_{i,x[\text{inp}(i)]}$, and one entry from \mathbf{t} .

The proof of Proposition 1 uses the following lemma:

Lemma 33. Fix $m, n, w \in \mathcal{N}$ and $\text{inp} : [m] \rightarrow [n]$. Let $\vec{S} = \text{SetSystem}(m, n, \text{inp}) = (\{S_{i,b}\}_{i \in [m], b \in \{0,1\}}, S_t)$, and let C be any weakly \vec{S} -respecting arithmetic circuit whose output wire is tagged with $T \subseteq [k]$. Then there exists a set $U \subseteq \{0, 1, *\}^m$ such that for every branching program BP of width w and length m on n input bits, with input tagging function inp , every prime p , and every $(\{\alpha_{i,b} \cdot \tilde{B}_{i,b}\}_{i \in [m], b \in \{0,1\}}, \mathbf{t}) \leftarrow \text{Rand}(BP, p)$,

(i)

$$C(\{\alpha_{i,b} \tilde{B}_{i,b}\}_{i \in [m], b \in \{0,1\}}, \mathbf{t}) \equiv \sum_{u \in U} C_u(\{\alpha_{i,b} \tilde{B}_{i,b}\}_{i \in [m], b \in \{0,1\}}, \mathbf{t})$$

where each C_u is a weakly \vec{S} -respecting arithmetic circuit, whose input wires are tagged only with sets $\in \{S_{i,u[i]}\}_{i \in [m]: u[i] \neq * } \cup \{S_t\}$, and whose output wire is tagged with T .

(ii) Each C_u above is the sum of several “monomial” circuits, where each monomial circuit performs only multiplications of elements in $(\{\alpha_{i,b} \cdot \tilde{B}_{i,b}\}_{i \in [m], b \in \{0,1\}}, \mathbf{t})$, is weakly \vec{S} -respecting, and has output wire tagged with T .

(iii) For each C_u above,

$$C_u(\{\alpha_{i,b} \tilde{B}_{i,b}\}_{i \in [m], b \in \{0,1\}}, \mathbf{t}) = \alpha_u \cdot p_u(\{\tilde{B}_{i,u[i]}\}_{i \in [m]: u[i] \neq *}, \mathbf{t})$$

where p_u is some polynomial, and $\alpha_u = (\prod_{i \in [m]: u[i] \neq *} \alpha_{i,u[i]})$. Furthermore, when p_u is viewed as a sum of monomials, each monomial contains exactly one entry from each $\tilde{B}_{i,u[i]}$ such that $u[i] \neq *$, and possibly one entry from \mathbf{t} . Further, p_u can be computed by a weakly \vec{S} -respecting circuit whose output wire is tagged with T .

The lemma can be proved using a simple induction. We provide a complete proof of the lemma in Section 3.8. Given this lemma, the proof of Proposition 1 is as follows:

Proof. **Part (i)** We consider the special case of Lemma 33 part (i), in which C is \vec{S} -respecting (as opposed to only weakly \vec{S} -respecting). In this case, we have that each C_u in the decomposition of C is also \vec{S} -respecting, and in particular, each C_u for $u \in U$ has its output wire tagged with the universe set $[k]$.

We first observe that for any C_u in the decomposition of C , u cannot contain $*$. This is because the output of C_u is tagged with $[k]$, and thus must have at least one input wire tagged with either of $S_{i,0}$ or $S_{i,1}$ for each i , or else the straddling set $\mathbb{S}^{\text{inp}(i)}$ will be incomplete, and thus the output wire cannot be tagged with $[k]$.

Further, we observe that for every $u \in U$, for every $j \in [n]$, there must be a bit $b_j \in \{0, 1\}$ such that for every $i \in [m]$ such that $\text{inp}(i) = j$, $u[i] = b_j$. This can be seen by considering any monomial circuit in C_u individually. Recall from Lemma 33

part (ii) that C_u is formed by summing some number of monomials circuits, each of which is \vec{S} -respecting and has output wire tagged with $[k]$. This means that $\mathbb{S}^j \subseteq [k]$ is covered by the elements of the monomial. However, since \mathbb{S}^j is constructed as a straddling set, the only way to cover \mathbb{S}^j in a monomial circuit that only contains multiplication gates, is by using either all sets from $\{S_{i,0} : \text{inp}(i) = j\}_{i \in m}$ or all sets from $\{S_{i,1} : \text{inp}(i) = j\}_{i \in m}$. This means, correspondingly, that u must be such that there is a bit $b_j \in \{0, 1\}$, for every $i \in [m]$ such that $\text{inp}(i) = j$, $u[i] = b_j$. Define $x \in \{0, 1\}^n$ so that $x[j] = b_j$ for all $j \in [n]$. In this way, we can define a one-to-one correspondence from each $u \in U$ to corresponding $x \in \{0, 1\}^n$, and we simply relabel each C_u to the corresponding C_x to get the desired decomposition of C . We observe that the additional conditions on each C_x can be achieved from the corresponding conditions on C_u as guaranteed by Lemma 33.

Part (ii) Part (ii) follows directly from Part (i) of this proposition, together with Lemma 33 part (iii), and the observation that each C_u in Lemma 33 is relabelled to C_x for some $x \in \{0, 1\}^n$ in Part (i) of this proposition. \square

Now we are ready to describe the simulator CSim. CSim gets as input 1^m , prime p , a \vec{S} -respecting circuit C , vectors v_0, v_1 and has oracle access to a length m branching program BP . Let X be the set of inputs and $\{p_x\}_{x \in X}$ be the single-input polynomials corresponding to the decomposition of C . For every $x \in X$, CSim queries BP on x , samples $d_x \leftarrow \text{KSim}(1^m, p, v_{BP(x)})$ and checks whether $p_x(d_x) = 0$. CSim outputs 1 if and only if for every input $x \in X$, $p_x(d_x) = 0$.

Now we prove correctness of our simulation. First, we prove some claims that will be useful. In each of these claims, let proj_x be defined with respect to the labeling function inp of the branching program BP . The following claim states

that if $C(\text{Rand}(BP, p))$ is always zero, then every single-input term is always zero.

Claim 34. *If $\Pr[C(\text{Rand}(BP, p)) = 0] = 1$ then for every input $x \in X$,*

$$\Pr[p_x(\text{proj}_x(\text{Rand}^B(BP, p))) = 0] = 1$$

Proof. Consider a fixed $d = (\{\tilde{B}_{i,b}\}_{i \in [m], b \in \{0,1\}}, \mathbf{t})$ in the support of $\text{Rand}^B(BP, p)$ and let $C_d(\{\alpha_{i,b}\}_{i \in [m], b \in \{0,1\}}) = C(\{\alpha_{i,b} \cdot \tilde{B}_{i,b}\}_{i \in [m], b \in \{0,1\}}, \mathbf{t})$. By Proposition 1, we know that

$$C_d(\{\alpha_{i,b}\}) = \sum_{x \in X} \left(\prod_{i \in [m]} \alpha_{i,x[\text{inp}(i)]} \right) p_x(\text{proj}_x(d))$$

and C_d is a degree $m + 2$ polynomial. By assumption, $C(\text{Rand}(BP, p))$ is always zero (over the support of $\text{Rand}(BP, p)$); hence, $C_d(\{\alpha_{i,b}\}) = 0$ for all non-zero $\{\alpha_{i,b}\}$. By the Schwartz-Zippel lemma, this can happen only if C_d is the zero polynomial. By the structure of C_d , this implies that for every $x \in X$, $p_x(\text{proj}_x(d)) = 0$. This argument works for every fixed value of d , hence we have that for every $x \in X$, $\Pr[p_x(\text{proj}_x(\text{Rand}^B(BP, p))) = 0] = 1$. \square

The next claim states that if $C(\text{Rand}(BP, p))$ is not always zero, then it is zero with small probability. Furthermore, there exists a single-input term that is zero with small probability.

Claim 35. *For any \vec{S} -respecting circuit C , if $\Pr[C(\text{Rand}(BP, p)) = 0] < 1$ then the following holds.*

1. $\Pr[C(\text{Rand}(BP, p)) = 0] \leq 16wm/p$
2. *There exists $x \in X$ such that $\Pr[p_x(\text{proj}_x(\text{Rand}^B(BP, p))) = 0] \leq 16wm/p$, where X is obtained from the decomposition of C by Proposition 1.*

Proof. We start by showing part 1.

Part 1: If $\text{Rand}(BP, p) = \text{Rand}^\alpha(\text{Rand}^B(BP, p))$ can be expressed as a low-degree ($\leq 2w$) polynomial on uniformly random values in \mathbb{Z}_p —namely, the α 's and the randomization matrices R_i 's—then by the Schwartz-Zippel lemma the first part of the claim directly follows. However, there are two barriers to applying this argument:

- Rand^B does not sample uniformly random matrices $\{R_i\}_{i \in [m]}$; rather, it chooses uniformly random *invertible* matrices R_i . Similarly, Rand^α does not sample uniformly random $\{\alpha_{i,b}\}_{i \in [m], b \in \{0,1\}}$; rather, it chooses uniformly random *non-zero* $\alpha_{i,b}$.
- Rand^B also needs to compute inverses R_i^{-1} to R_i for every $i \in [m]$ (which may no longer be expressed as low degree polynomials in the matrices $\{R_i\}_{i \in [m]}$).

To handle the second issue, consider the distribution $\text{Rand}_{adj}^B(BP, p)$ that is defined exactly as $\text{Rand}^B(BP, p)$ except that for every $i \in [m]$ it uses $adj(R_i) = R_i^{-1} \det(R_i)$ instead of R_i^{-1} . Note that every entry of the adjoint of a $w \times w$ matrix M is some cofactor of M (obtained by the determinant of the $w - 1 \times w - 1$ matrix obtained by deleting some row and column of A). Hence every entry of $adj(R_i)$ can be expressed as a degree w polynomial in R_i . Let $\text{Rand}_{adj}(BP, p) = \text{Rand}^\alpha(\text{Rand}_{adj}^B(BP, p))$. It follows that $\text{Rand}_{adj}(BP, p)$ is computed by degree (at most) $2w$ polynomial in the matrices $\{R_i\}_{i \in [m]}$ and scalars $\{\alpha_{i,b}\}_{i \in [m], b \in \{0,1\}}$.

Furthermore, we show that $\Pr[C(\text{Rand}_{adj}(BP, p)) = 0] = \Pr[C(\text{Rand}(BP, p)) = 0]$. Recall that by Proposition 1,

$$C \equiv \sum_{x \in X} C_x$$

and for each C_x above and every $(\{\alpha_{i,b} \cdot \tilde{B}_{i,b}\}_{i \in [m], b \in \{0,1\}}, \mathbf{t}) \leftarrow \text{Rand}(BP, p)$,

$$C_x(\{\alpha_{i,b} \cdot \tilde{B}_{i,b}\}_{i \in [m], b \in \{0,1\}}, \mathbf{t}) = \alpha_x \cdot p_x(\{\tilde{B}_{i,x[\text{inp}(i)]}\}_{i \in [m]}, \mathbf{t})$$

where $\alpha_x = (\prod_{i \in [m]} \alpha_{i,x[\text{inp}(i)]})$ and p_x is a polynomial such that, when viewed as a sum of monomials, each monomial contains exactly one entry from each $\tilde{B}_{i,x[\text{inp}(i)]}$, and one entry from \mathbf{t} . Recall that for every $i \in [m]$,

$$\tilde{B}_{i,x[\text{inp}(i)]} = R_{i-1} B_{i,x[\text{inp}(i)]} R_i^{-1}$$

For every $i \in [m]$, replacing R_i^{-1} with $\text{adj}(R_i)$ has the effect of multiplying each monomial in p_x with the scalar $\det(R_i)$. Hence

$$C_x(\text{Rand}_{\text{adj}}(BP, p)) = \left(\prod_{i \in [m]} \det(R_i) \right) \cdot C_x(\text{Rand}(BP, p))$$

Since C is the sum of such C_x terms, it holds that $C(\text{Rand}_{\text{adj}}(BP, p)) = (\prod_{i \in [m]} \det(R_i)) C(\text{Rand}(BP, p))$. For every $i \in [m]$, by invertibility, $\det(R_i) \neq 0$ and hence

$$\Pr[C(\text{Rand}_{\text{adj}}(BP, p)) = 0] = \Pr[C(\text{Rand}(BP, p)) = 0]$$

So far, we have that $\text{Rand}_{\text{adj}}(BP, p)$ is computed by a degree $2w$ polynomial in the matrices $\{R_i\}_{i \in [m]}$ and scalars $\{\alpha_{i,b}\}_{i \in [m], b \in \{0,1\}}$. However the first issue remains: each R_i is uniformly random invertible and each $\alpha_{i,b}$ is uniformly random non-zero, whereas we need them to be uniformly random. Consider the distribution $\text{Rand}_{\text{adj},U}(BP, p)$ that is obtained by the computing the same polynomial on uniformly random matrices $\{R_i\}_{i \in [m]}$ and scalars $\{\alpha_{i,b}\}_{i \in [m], b \in \{0,1\}}$ over \mathbb{Z}_p . In Claim 44, we show that the statistical distance between $\text{Rand}_{\text{adj}}(BP, p)$ and $\text{Rand}_{\text{adj},U}(BP, p)$ is at most $8wm/p$. Furthermore, the support of $\text{Rand}_{\text{adj},U}(BP, p)$ contains the support of $\text{Rand}_{\text{adj}}(BP, p)$. This implies that if $\Pr[C(\text{Rand}_{\text{adj}}(BP, p)) = 0] < 1$ then $\Pr[C(\text{Rand}_{\text{adj},U}(BP, p)) = 0] < 1$.

We now turn to proving the statement of the claim. Using facts shown above, we have that

$$\begin{aligned} \Pr[C(\text{Rand}(BP, p)) = 0] < 1 &\implies \Pr[C(\text{Rand}_{adj}(BP, p)) = 0] < 1 \\ &\implies \Pr[C(\text{Rand}_{adj,U}(BP, p)) = 0] < 1 \end{aligned}$$

By Proposition 1, C evaluates a $m + 1$ degree polynomial, and $\text{Rand}_{adj,U}(BP, p)$ is computed by a degree $2w$ polynomial in uniformly random values in \mathbb{Z}_p . By the Schwartz-Zippel lemma,

$$\begin{aligned} \Pr[C(\text{Rand}_{adj,U}(BP, p)) = 0] < 1 \\ \implies \Pr[C(\text{Rand}_{adj,U}(BP, p)) = 0] \leq 2w(m + 1)/p \leq 8wm/p \end{aligned}$$

We have that the statistical distance between $\text{Rand}_{adj,U}(BP, p)$ and $\text{Rand}_{adj}(BP, p)$ is at most $8wm/p$. Therefore, $\Pr[C(\text{Rand}(BP, p)) = 0] = \Pr[C(\text{Rand}_{adj}(BP, p)) = 0] \leq 16wm/p$ thus proving the first part of the claim. We proceed to show part 2.

Part 2: By Proposition 1, for every $x \in X$, there exists a \vec{S} -respecting arithmetic circuit C_x such that for every $(\{\alpha_{i,b} \cdot \tilde{B}_{i,b}\}_{i \in [m], b \in \{0,1\}}, \mathbf{t}) \leftarrow \text{Rand}(BP, p)$,

$$C_x(\{\alpha_{i,b} \cdot \tilde{B}_{i,b}\}_{i \in [m], b \in \{0,1\}}, \mathbf{t}) = \alpha_x \cdot p_x(\{\tilde{B}_{i,x[\text{inp}(i)]}\}_{i \in [m]}, \mathbf{t})$$

where $\alpha_x = (\prod_{i \in [m]} \alpha_{i,x[\text{inp}(i)]})$ and $C = \sum_{x \in X} C_x$. In particular, $p_x(\{\tilde{B}_{i,x[\text{inp}(i)]}\}_{i \in [m]}, \mathbf{t}) = 0$ iff $C_x(\{\alpha_{i,b} \cdot \tilde{B}_{i,b}\}_{i \in [m], b \in \{0,1\}}, \mathbf{t}) = 0$ (since $\alpha_{i,b}$ is non-zero).

Thus, we have that

$$\begin{aligned} \Pr[C(\text{Rand}(BP, p)) = 0] &= \Pr[C_x(\text{Rand}^\alpha(\text{Rand}^B(BP, p))) = 0] \\ &= \Pr[p_x(\text{proj}_x(\text{Rand}^B(BP, p))) = 0] \end{aligned}$$

There must exist an input $x \in X$ such that $\Pr[C_x(\text{Rand}(BP, p)) = 0] < 1$ or else $\Pr[C(\text{Rand}(BP, p)) = 0] = 1$. By the first part of the claim, it follows that

$$\Pr[C(\text{Rand}(BP, p)) = 0] \leq 16wm/p,$$

which concludes the proof. \square

Now we analyze the correctness of the simulator CSim. We consider the following two cases: when $C(\text{Rand}(BP, p))$ is always zero, and otherwise.

Case 1: $Pr[C(\text{Rand}(BP, p)) = 0] = 1$: In this case we will show that the simulation always succeeds. If $Pr[C(\text{Rand}(BP, p)) = 0] = 1$ then by Claim 34, for every $x \in X$, $Pr[p_x(\text{proj}_x(\text{Rand}^B(BP, p))) = 0] = 1$. Recall that $\text{KSim}(1^m, p, v_{BP(x)})$ simulates $\text{proj}_x(\text{Rand}^B(BP, p))$ perfectly. Therefore, CSim always outputs 1 and hence succeeds.

Case 2: $Pr[C(\text{Rand}(BP, p)) = 0] < 1$: In this case, by the first part of Claim 35 we have that

$$Pr[\text{isZero}(C(\text{Rand}(BP, p))) = 1] \leq 16wm/p$$

By the perfect simulation of KSim, we have that

$$Pr[\text{CSim}^{BP} = 1] = Pr[\forall x (d_x \leftarrow \text{proj}_x(\text{Rand}^B(BP, p)) : p_x(d_x) = 0)]$$

By second part of Claim 35 there exists input x_C such that $Pr[p_{x_C}(\text{proj}_{x_C}(\text{Rand}^B(BP, p))) = 0] \leq 16wm/p$. Therefore,

$$Pr[\text{CSim}^{BP} = 1] \leq Pr[p_{x_C}(\text{proj}_{x_C}(\text{Rand}^B(BP, p))) = 0] \leq 16wm/p$$

Therefore, by a union bound we have that

$$Pr[\text{isZero}(C(\mathcal{D})) = \text{CSim}^{BP} = 0] > 1 - 32wm/p$$

This concludes the proof of the lemma. \square

\square

Restricting to Entropic Message Samplers

We here show that the message sampler M in the previous section satisfies the required high-entropy condition (required by the notion of entropic semantical security); that is, M is entropically valid.

Recall that the message sampler M in the proof of Theorem 31 gets as input the description of a ring $R = \mathbb{Z}_p$ and samples $(\vec{m}_0, \vec{m}_1, \vec{z})$ such that (\vec{m}_0, \vec{z}) and (\vec{m}_1, \vec{z}) are the “randomizations” (as defined in the description of Rand) of fixed branching programs. We now show the following proposition, which combined with the fact that the length m of the branching programs is polynomial in $\log |R|$ (recall that $R = \mathbb{Z}_p$ where p is a prime exponential in the multilinearity parameter k which is $< 3m$), implies that the output of a non-terminal set-respecting circuit on input (\vec{m}_b, \vec{z}) (for both $b \in \{0, 1\}$) has min-entropy $\log |R| - O(\log \log |R|)$, as required.

Proposition 2. *Let BP be a branching program of length m , width w , input length n and input labeling function inp . Let p be a prime and $\vec{S} = \text{SetSystem}(m, n, \text{inp})$. Let C be a non-terminal \vec{S} -respecting arithmetic circuit that computes a non-zero polynomial. Then we have that*

$$H_\infty(C(\text{Rand}(BP, p))) \geq \log\left(\frac{P}{12wm}\right)$$

or equivalently, for any fixed output $a \in \mathbb{Z}_p$

$$\Pr[C(\text{Rand}(BP, p)) = a] \leq 12wm/p$$

Proof. Let T be the set that tags the output wire of C as per the construction given in Definition 32. Since C is non-terminal \vec{S} -respecting, we have that T is a strict subset of $[k]$ where $(k, \vec{S}) = \text{SetSystem}(m, n, \text{inp})$. By Lemma 33 part (iii), there

exists a set U of labels $u \in \{0, 1, *\}$ such that for every $(\{\alpha_{j,b} \cdot \tilde{B}_{j,b}\}_{j \in [m], b \in \{0,1\}}, \mathbf{t}) \leftarrow \text{Rand}(BP, p)$ we have that

$$C(\{\alpha_{j,b} \cdot \tilde{B}_{j,b}\}_{j \in [m], b \in \{0,1\}}, \mathbf{t}) = \sum_{u \in U} \alpha_u \cdot p_u(\{\tilde{B}_{j,u[j]}\}_{j \in [m]: u[j] \neq *}, \mathbf{t}) \quad (3.4)$$

where $\alpha_u = \prod_{j \in [m]: u[j] \neq *} \alpha_{j,u[j]}$. Furthermore, each p_u is computed by a weakly \vec{S} -respecting circuit whose output wire is also tagged with T . Since C computes a non-zero polynomial, there must exist $v \in U$ such that p_v is a non-zero polynomial. We now have the following claim.

Claim 36. $Pr[p_v(\{\tilde{B}_{j,v[j]}\}_{j \in [m]: v[j] \neq *}, \mathbf{t}) = 0] \leq 10wm/p$.

Proof. To see this, we first observe that since T is a strict subset of $[k]$ and p_v is computed by a \vec{S} -respecting circuit whose output wire is tagged with T , either p_v does not operate on some level of the branching program or it does not operate on \mathbf{t} ; that is, either,

- there exists $j \in [m]$ such that $v[j] = *$, or
- p_v is not a function of \mathbf{t} .

In the first case, by an argument similar to that in Claim 29, we can show that the distribution $(\{\tilde{B}_{j,v[j]}\}_{j \in [m]: v[j] \neq *}, \mathbf{t})$ is identical to the distribution $(\{R_j\}_{j \in [m]: v[j] \neq *}, \text{col}_1(R_{m+1}))$ where $\{R_j\}_{j=1}^{m+1}$ are random invertible matrices over $\mathbb{Z}_p^{w \times w}$. By Claim 44, this distribution is statistically $8wm/p$ -close to the distribution where each matrix entry is uniformly random in \mathbb{Z}_p . Furthermore, since p_v is computed by a \vec{S} -respecting circuit, it is of degree at most $m + 1 < 2wm$. By the Schwartz Zippel lemma, the evaluation of p_v on such random inputs from \mathbb{Z}_p is zero with probability at most $2wm/p$. All in all, we have $Pr[p_v(\{\tilde{B}_{j,v[j]}\}_{j \in [m]: v[j] \neq *}, \mathbf{t}) = 0] \leq 10wm/p$.

In the second case, p_v acts on the $\{\tilde{B}_{j,v[j]}\}_{j \in [m]}$. Following Claim 29, this distribution is identical to that of m random invertible matrices over $\mathbb{Z}_p^{w \times w}$. Similarly to the first case, it follows that $Pr[p_v(\{\tilde{B}_{j,v[j]}\}_{j \in [m]}) = 0] \leq 10wm/p$. \square

Let E be the event that $p_v(\{\tilde{B}_{j,v[j]}\}_{j \in [m]:v[j] \neq *}, \mathbf{t}) \neq 0$. For any fixed output $a \in \mathbb{Z}_p$ we have that

$$Pr[C(\text{Rand}(BP, p)) = a] \leq Pr[C(\text{Rand}(BP, p)) = a|E] + Pr[\bar{E}] \quad (3.5)$$

For a fixed $\{\tilde{B}_{j,b}\}_{j \in [m], b \in \{0,1\}}$ let $q_{(\tilde{B},a)}$ be a polynomial in variables $\{\alpha_{j,b}\}_{j \in [m], b \in \{0,1\}}$ such that

$$q_{(\tilde{B},a)}(\{\alpha_{j,b}\}_{j \in [m], b \in \{0,1\}}) = C(\{\alpha_{j,b} \cdot \tilde{B}_{j,b}\}_{j \in [m], b \in \{0,1\}}) - a$$

When the event E occurs, we claim that the resulting polynomial $q_{(\tilde{B},a)}$ is a non-zero polynomial of degree at most m . This can be easily seen given the decomposition of C in (3.4). When $q_{(\tilde{B},a)}$ is a non-zero polynomial then by the Schwartz Zippel lemma, its evaluation on uniformly random non-zero inputs $\{\alpha_{j,b}\}_{j \in [m], b \in \{0,1\}}$ is zero with probability at most $m/p - 1 \leq 2wm/p$. Therefore, we have

$$Pr[C(\text{Rand}(BP, p)) = a|E] = Pr[q_{\tilde{B}}(\{\alpha_{j,b}\}) = 0|E] \leq \frac{2wm}{p} \quad (3.6)$$

Combining (3.6) and (3.5) and Claim 36, we have $Pr[C(\text{Rand}(BP, p)) = a] \leq 12wm/p$. \square

3.4.4 Achieving Obfuscation for Arbitrary Programs

[57] show that any indistinguishability obfuscation scheme for NC^1 can be bootstrapped into an indistinguishability obfuscation scheme for all poly-sized cir-

circuits using FHE. That is, they prove the following theorem.

Theorem 37 ([57]). *Assume the existence of indistinguishability obfuscators iO for NC^1 and a leveled Fully Homomorphic Encryption scheme with decryption in NC^1 . Then there exists an indistinguishability obfuscator iO' for $P/poly$.*

Applying their construction to our indistinguishability obfuscator yields an indistinguishability obfuscator for arbitrary polynomial size circuits:

Theorem 38. *Assume the existence of an entropic semantically secure multilinear encoding scheme and a leveled Fully Homomorphic Encryption scheme with decryption in NC^1 . Then there exists indistinguishability obfuscators for $P/poly$.*

3.5 iO from Single-Distribution Semantical Security

The assumption that a scheme satisfies semantical security w.r.t. some class of message samplers may perhaps be best viewed as a *class of assumptions* (or a “meta-assumption”, just like the “uber assumption” of [25]), or alternatively as an *interactive assumption*, where the attacker first selects the sets \vec{S}, \vec{T} and the message sampler M , and then gets a challenge according to the message sampler.

This view point also clarifies that even for the above-mentioned restricted classes of message distributions, semantical security is not an *efficiently falsifiable* assumption [99]: the problem is that there may not exist an efficient way of checking whether a message sampler is valid (which requires checking that all set-respecting circuits are constant with overwhelming probability).

We here show that a single, falsifiable, instance of this class of assumptions suffices for proving security of indistinguishability obfuscator, albeit at the cost of subexponential hardness.

3.5.1 Single-Distribution Semantical Security

Let us start by formalizing a “single-distribution” version of semantical security, where we restrict semantical security to hold w.r.t. to a single *efficiently samplable* distribution over pairs of message samplers M , and sets \vec{S}, \vec{T} . We call this distribution over message samplers and sets an *instance sampler*. Analogously to the notion of a valid message sampler, we now define a notion of a *valid instance sampler* as follows:

Definition 43. *We say that a PPT Sam is a (c, q) -(entropically) valid instance sampler if*

- *There exist a polynomial $k(\cdot)$, such that for every $n \in \mathbb{N}$, for every $r_n \in \{0, 1\}^\infty$, $\text{Sam}(1^n, r_n)$ outputs a tuple $(\vec{S}_n, \vec{T}_n, M_n)$, where \vec{S}_n, \vec{T}_n are sequences of sets over $[k(n)]$ with $|\vec{S}_n| = c(k(n))$ and $|\vec{T}_n| = q(k(n))$.*
- *For every sequence of random tapes $\{r_n\}_{n \in \mathbb{N}}$, $\{M_n\}_{n \in \mathbb{N}}$ is (entropically) $\{\vec{S}_n, \vec{T}_n\}_{n \in \mathbb{N}}$ -respecting, where for every $n \in \mathbb{N}$, $(\vec{S}_n, \vec{T}_n, M_n) \leftarrow \text{Sam}(1^n; r_n)$.*

Definition 44 (Single-distribution Semantic Security). *Let \mathcal{E} be a graded encoding scheme and Sam be a (c, q) -valid instance sampler. We say that \mathcal{E} is semantically secure w.r.t. Sam if for every nuPPT adversary A , there exists a negligible function ϵ such that for every security parameter $n \in \mathbb{N}$,*

$$|\Pr[\text{Output}'_0(1^n) = 1] - \Pr[\text{Output}'_1(1^n) = 1]| \leq \epsilon(n)$$

where $\mathbf{Output}'_b(1^n)$ is A 's output in the following game:

- Let $\vec{S}_n, \vec{T}_n, M_n \leftarrow \mathbf{Sam}(1^n)$.
- Let k_n be such that \vec{S}_n and \vec{T}_n are sequences of sets over $[k_n]$. Let $(\mathbf{sp}, \mathbf{pp}) \leftarrow \mathbf{InstGen}(1^n, 1^{k_n})$.
- Let $\vec{m}_0, \vec{m}_1, \vec{z} \leftarrow M_n(1^n, \mathbf{pp})$.
- Let $\vec{u}_b \leftarrow \{\mathbf{Enc}(\mathbf{sp}, \vec{m}_0[i], \vec{S}_n[i])\}_{i=1}^{c(n)}, \{\mathbf{Enc}(\mathbf{sp}, \vec{z}[i], \vec{T}_n[i])\}_{i=1}^{q(n)}$.
- Finally, run $A(1^n, \mathbf{pp}, (\vec{S}_n, \vec{T}_n), M_n, \vec{u}_b)$.

Note that given an $(O(1), O(k))$ -valid instance sampler \mathbf{Sam} , the assumption that \mathcal{E} is semantically-secure w.r.t. \mathbf{Sam} is a special case of the assumption that \mathcal{E} is (constant-message) semantically secure; if \mathcal{E} is not semantically secure w.r.t. \mathbf{Sam} , there exists ensembles $\{r_n\}_{n \in \mathcal{N}}$, $\{\vec{S}_n, \vec{T}_n\}_{n \in \mathcal{N}}$ and $\{M_n\}_{n \in \mathcal{N}}$ such that $\vec{S}_n, \vec{T}_n, M_n = \mathbf{Sam}(1^n; r_n)$ (and thus $\{M_n\}_{n \in \mathcal{N}}$ is a valid message sampler for $\{\vec{S}_n, \vec{T}_n\}_{n \in \mathcal{N}}$, yet the nuPPT $A(1^n, \cdot, \vec{S}_n, \vec{T}_n, M_n, \cdot)$ breaks semantical security when considering $\{\vec{S}_n, \vec{T}_n\}_{n \in \mathcal{N}}$ and $\{M_n\}_{n \in \mathcal{N}}$.

Furthermore, that given an $(O(1), O(k))$ -(entropically) valid instance sampler \mathbf{Sam} , the assumption that \mathcal{E} is semantically-secure w.r.t. \mathbf{Sam} is a non-interactive and efficiently falsifiable (decisional) assumption—in essence, it is a specific instance of a DDH-type assumption over multilinear encodings.

3.5.2 Basing Security on Single-Distribution Semantical Security

We now show how to slightly modify the construction iO from Section 3.4.3 so that we can base it on single-distribution semantical security assumption. This time, however, we require subexponentially-hard semantical security (and as such the assumption is incomparable to the one needed for the scheme from Section 3.4.3.)

Towards this, we introduce a new notion of *neighboring-input indistinguishability obfuscation*. As we shall see, the assumption that a scheme satisfies neighboring-input iO is already an efficiently falsifiable assumption. We then show that a) *exponentially-secure* neighboring-input iO implies “full” iO , and b) exponentially-secure neighboring-input iO can be based on subexponentially-hard single-distribution semantic security. (We mention a very recent work by Gentry, Lewko and Waters [65] in the context of *witness encryption* [59] that similarly defines a falsifiable primitive “positional witness encryption” that implies the full-fledged notion with an exponential security loss.)

Neighboring-input Indistinguishability Obfuscation

We start by recall a different “merge” procedure from the work of Boyle, Chung and Pass [32]: Given two NC^1 circuits C_0, C_1 taking (at most) n -bit inputs, and a string z , let $\widehat{\text{Merge}}(C_0, C_1, z)$ be a circuit that on input x runs $C_0(x)$ if $x \geq z$ and $C_1(x)$ otherwise. ([32] use this type of merged circuits to perform a binary search and prove that indistinguishability obfuscation implies differing-input obfuscation for circuits that differ in only polynomially many inputs.) Also,

$\widehat{\text{Merge}}$ is defined such that $\widehat{\text{Merge}}(C_0, C_1, 0) = C_0$ and $\widehat{\text{Merge}}(C_0, C_1, 2^n) = C_1$. It is easy to see that an NC^1 circuit computing $\widehat{\text{Merge}}(C_0, C_1, z)$ can be efficiently found given NC^1 circuits C_0, C_1 and z ; (abusing notation) let $\widehat{\text{Merge}}$ denote an efficient procedure that outputs such a circuit.

The notion of neighboring-input iO relaxes iO by only requiring that security holds with respect to “neighboring-input” programs $\widehat{\text{Merge}}(C_0, C_1, z)$, $\widehat{\text{Merge}}(C_0, C_1, z + 1)$ that are functionally equivalent. Note that checking whether $\widehat{\text{Merge}}(C_0, C_1, z)$, $\widehat{\text{Merge}}(C_0, C_1, z + 1)$ are functionally equivalent is easy: they are equivalent iff $C_0(z) = C_1(z)$. As such, the assumption that a scheme satisfies neighboring-input iO is efficiently falsifiable.

Definition 45. *A uniform PPT machine iO is a neighboring-input indistinguishability obfuscator for the class of circuits $\{C_n\}_{n \in \mathbb{N}}$ if it satisfies the same correctness condition as in Definition 24 but the security condition is replaced by:*

- **Security:** *For every nuPPT adversary A there exists a negligible function ϵ such that for all $n \in \mathbb{N}$, all $C_0, C_1 \in \mathcal{C}_n^1$ and all $z \in \{0, 1\}$ such that $C_0(z) = C_1(z)$,*

$$|\Pr[A(1^n, C'_0, C'_1, z, iO(1^n, C'_0)) = 1] - \Pr[A(1^n, C'_0, C'_1, z, iO(1^n, C'_1)) = 1]| \leq \epsilon(n)$$

where $C'_b = \widehat{\text{Merge}}(C_0, C_1, z + b)$.

We additionally say that iO is exponentially-secure if for every nuPPT A the above indistinguishability gap is bounded by $\epsilon(n) = 2^{-O(n^2)}$.

Theorem 39. *There exists an $(O(1), O(k))$ -entropically valid instance sampler Sam , such that if there exists an encoding scheme that is subexponentially-hard semantically secure w.r.t. Sam , then there exists an exponentially-secure neighboring-input indistinguishability obfuscator for \mathcal{C}^1 .*

Proof. Consider the obfuscator $i\mathcal{O}(\cdot, \cdot, \cdot)$ for NC^1 presented in Section 3.4.3. We change it to run the underlying multilinear encoding scheme with security parameter $n' = n^{2/\alpha}$, where α is the subexponential security constant for the encoding scheme. Let c^* be the constant such that the sizes and depth of $\widehat{\text{Merge}}(C_0, C_1, z)$ where $C_0, C_1 \in C_n^1$ and $z \in \{0, 1\}^n$ are bounded by n^{c^*} and $c^* \log(n)$ respectively. We show that $i\mathcal{O}(c^*, \cdot, \cdot) = i\mathcal{O}(\cdot, \cdot)$ is an exponentially-secure indistinguishability obfuscator for C^1 based on subexponentially-hard semantical security with respect to an instance sampler Sam .

Assume for contradiction there exists nuPPT A such that for infinitely many n , there exist $C_0, C_1 \in C_n^1, z \in \{0, 1\}^n$ such that $C_0(z) = C_1(z)$ and A given $(1^n, C'_0, C'_1, z)$ where $C'_0 = \widehat{\text{Merge}}(C_0, C_1, z)$ and $C'_1 = \widehat{\text{Merge}}(C_0, C_1, z+1)$, distinguishes $i\mathcal{O}(1^n, C'_0)$ and $i\mathcal{O}(1^n, C'_1)$, with probability, say, 2^{-n^2} .

We define hybrid distributions similarly as in the proof in Section 3.4.2 corresponding to $i\mathcal{O}(1^n, C'_0)$ and $i\mathcal{O}(1^n, C'_1)$. Recall that each of these hybrids correspond to one step in the transition from a branching program for C'_0 to a branching program C'_1 , where each step changes at most two levels of the branching program. Let $h(n)$ be the number of such hybrids. We have that the circuits C'_0 and C'_1 determine for every $j \in [h(n) - 1]$ a hybrid distribution H_j such that H_0 is identical to $i\mathcal{O}(1^n, C'_0)$, $H_{h(n)}$ is identical to $i\mathcal{O}(1^n, C'_1)$ and for every $j \in [h(n) - 1]$, indistinguishability of H_j and H_{j+1} follows from neighboring-matrix indistinguishability obfuscation which in turn follows from a reduction to semantic security.

We now define $\text{Sam}(1^{n'}; r_{n'})$ as follows: Using random coins $r_{n'}$, Sam uniformly samples $C_0, C_1 \leftarrow C_n^1, z \leftarrow \{0, 1\}^n$ and a random hybrid index $j \in [h(n) - 1]$. It checks whether $C_0(z) = C_1(z)$ and if not, it sets $C_1 = C_0$. Next, it generates

$C'_0 = \widehat{\text{Merge}}(C_0, C_1, z)$ and $C'_1 = \widehat{\text{Merge}}(C_0, C_1, z + 1)$. Finally, it outputs the sets $(\vec{S}_{n'}, \vec{T}_{n'})$ and message sampler $M_{n'}$ used in the reduction to semantic security when arguing indistinguishability of hybrids H_j and H_{j+1} , as determined by the circuits C'_0 and C'_1 .

Note that since the pair of the circuits C'_0, C'_1 sampled by Sam are always functionally equivalent, by the same proof as in Section 3.4.3 (more specifically, Lemma 32), we have that the messages $\vec{m}_0, \vec{m}_1, \vec{z}$ output by $M_{n'}$ are such that every $(\vec{S}_{n'}, \vec{T}_{n'})$ -respecting circuit is constant on both \vec{m}_0, \vec{z} and \vec{m}_1, \vec{z} , except with probability at most $Q(n', k)/|R|$ for some fixed polynomial $Q(\cdot, \cdot)$. Thus, for every sequence of random tapes $\{r_n\}_{n \in \mathcal{N}}, \{M_n\}_{n \in \mathcal{N}}$ is $\{\vec{S}_n, \vec{T}_n\}_{n \in \mathcal{N}}$ -respecting, where for every $n \in \mathcal{N}$, $\vec{S}_n, \vec{T}_n, M_n = \text{Sam}(1^n; r_n)$. We conclude that Sam is a $(O(1), O(k))$ -valid instance sampler.

By assumption, there exists a $j \in [h(n) - 1]$ such that A distinguishes H_j and H_{j+1} with advantage $2^{-n^2}/h(n)$. We now define a nuPPT attacker A' for semantical security w.r.t. Sam: For each n' , A' receives as non-uniform advice the index j^* and proceeds as follows: $A'(1^{n'}, \text{pp}, (\vec{S}_{n'}, \vec{T}_{n'}), M_{n'}, \vec{u}_b)$ examines $M_{n'}$ and extracts the underlying circuits C_0^*, C_1^* the underlying merge index z^* and the underlying hybrid index j^* from it. (We assume $M_{n'}$ is defined so that this information is efficiently extractable.) If $j = j^*$, $C_0^* = C'_0, C_1^* = C'_1$ and $z^* = z$, A' executes $A(1^n, C_0^*, C_1^*, z^*, (\text{pp}, \vec{u}_b))$, and otherwise simply outputs 1.

Let us now analyze the success probability of A' :

- Conditioned on the event when $j = j^*$, $C_0^* = C'_0, C_1^* = C'_1$ and $z^* = z$, A' distinguishes with advantage $2^{-n^2}/h(n)$.
- Otherwise A' 's output is 1.

Since C_0^*, C_1^*, z^*, j^* are chosen at random, it follows that A' has a total distinguishing advantage of at least $2^{-3n} \cdot 2^{-n^2} / h(n)^2 = 2^{-O(n^2)} = 2^{-O(n^\alpha)}$, which contradicts the assumption that the encoding scheme is subexponentially secure with respect to Sam. \square

From $ni-iO$ to iO

Theorem 40. *If there exists PPT iO that is an exponentially-secure neighboring-input indistinguishability obfuscator for C^1 , then there exists a PPT iO' that is a subexponentially-secure indistinguishability obfuscator for NC^1 .*

Proof. Assume the existence of a PPT iO that is an exponentially-secure neighboring-input indistinguishability obfuscator for the class C^1 . We show that iO is a (subexponentially-secure) indistinguishability obfuscator for C^1 ; by Lemma 19, this suffices for concluding the existence of (subexponentially-secure) indistinguishability obfuscators for NC^1 .

Assume there exists some nuPPT A such that for infinitely many n , there exists a pair of functionally equivalent circuits $C_n^0, C_n^1 \in C_n^1$ such that A distinguishes $iO(1^n, C_n^0)$ and $iO(1^n, C_n^1)$ with probability, say, 2^{-n} . For any such n , consider a sequence of $2^n + 1$ hybrid distributions, where

- $H_0 = iO(1^n, C_n^0) = iO(1^n, \widehat{\text{Merge}}(C_n^0, C_n^1, 0))$
- $H_i = iO(1^n, \widehat{\text{Merge}}(C_n^0, C_n^1, i))$ for $i \in [1, \dots, 2^n - 1]$
- $H_{2^n} = iO(1^n, C_n^1) = iO(1^n, \widehat{\text{Merge}}(C_n^0, C_n^1, 2^n))$

There must exist some z such that A distinguishes H_z and H_{z+1} with advantage at least $2^{-n} \cdot 2^{-n} = 2^{-2n}$. Thus, there exists some sequence of programs $\{C_n^0, C_n^1\}_{n \in \mathcal{N}}$

where $C_n^0, C_n^1 \in \mathcal{C}_n^1$ and a sequence of inputs $\{z_n\}_{n \in \mathcal{N}}$, $z_n \in [0, \dots, 2^n - 1]$, such that for infinitely many n , A distinguishes $i\mathcal{O}(1^n, \widehat{\text{Merge}}(C_n^0, C_n^1, z_n))$ and $i\mathcal{O}(1^n, \widehat{\text{Merge}}(C_n^0, C_n^1, z_n + 1))$ with advantage 2^{-2n} . This directly contradicts the exponential security of the neighboring-input indistinguishability obfuscator $i\mathcal{O}$. \square

Combing the above theorems, we get the following corollary.

Theorem 41. *There exists an $(O(1), O(k))$ -entropically valid instance sampler Sam , such that if there exists an encoding scheme that is subexponentially-hard semantically secure w.r.t. Sam , then there exists a subexponentially-secure indistinguishability obfuscator for NC^1 .*

3.6 Alternative Security Notions of Semantical Security Encodings

In this section we consider alternative ways of defining security of multilinear encodings. First, in section 3.6.1 we show that semantical security holds (in a very strong sense) w.r.t. generic attackers. Next, in section 3.6.2 we consider various “uber assumptions” (similar to the uber-assumption of [25] in the context of bilinear maps)²⁸ which capture the intuition that “if an algebraic decisional assumption holds w.r.t. to generic attacks, then it also holds with respect to nuPPT attackers”. As we shall see the perhaps most natural formalization of this notion is *false* (under standard cryptographic assumptions)—in particular, we give a concrete example of a algebraic decisional assumption that holds in

²⁸We thank Shai Halevi for pointing out the connection with [25].

the generic model but is false w.r.t. nuPPT attackers. We finally consider alternative ways for formalizing such an uber assumption.

3.6.1 Semantical Security w.r.t. Algebraic Attackers

We begin by showing that semantic security holds in the generic model. We formally define an *algebraic adversary* or *generic adversary* by considering adversaries that interact with the following oracle.

Definition 46 (Oracle \mathcal{M}). *Let \mathcal{M} be an oracle which operates as follows:*

- \mathcal{M} gets as initial input a ring R , $k \in \mathbb{N}$ and list L of m pairs $\{(\alpha_i, S_i)\}_{i=1}^m$, $\alpha \in R$ and $S \subseteq [k]$.
- Every oracle query to \mathcal{M} is an arithmetic circuit $C : R^m \rightarrow R$. When queried with C , \mathcal{M} checks whether C is a \vec{S} -respecting arithmetic circuit where $\vec{S} = \{S_i\}_{i=1}^m$. If not, \mathcal{M} outputs \perp . Otherwise, \mathcal{M} computes C on $\{\alpha_i\}_{i=1}^m$ and outputs 1 if and only if the output of C is zero, and outputs 0 otherwise.

To formalize that (even subexponentially-hard) semantical security holds w.r.t. generic attackers, we define a stronger notion of a set-respecting message samplers—which requires not only that the output of every set-respecting circuit is constant with overwhelming probability, but also that this holds for the output of any *unbounded* algebraic attacker that is restricted to *polynomially-many* zero-test queries—and show that this notion in fact already is implied by the standard one. This shows that semantical security holds in a very strong sense w.r.t. to generic attackers.

Definition 47 (Strongly Respecting Message Sampler). *We say that a nuPPT M is a strongly $\{(\vec{S}_n, \vec{T}_n)\}_{n \in \mathbb{N}}$ -respecting message sampler (or strongly valid w.r.t. $\{(\vec{S}_n, \vec{T}_n)\}_{n \in \mathbb{N}}$) if it satisfies the same conditions as in Definition 34 but where the second bullet is replaced by the following:*

- *For every polynomial p , there exists some polynomial Q such that for every $n \in \mathbb{N}$, every (sp, pp) in the support of $\text{InstGen}(1^n, 1^{k_n})$, every (deterministic) oracle algorithm A that on input 1^n makes at most $p(n)$ oracle queries, there exists some string $\alpha \in \{0, 1\}^*$ such that*

$$\Pr[(\vec{m}_0, \vec{m}_1, \vec{z}) \leftarrow M(1^n, \text{pp}) : A^{\mathcal{M}(\text{pp}, \vec{p}_0)}(1^n) = A^{\mathcal{M}(\text{pp}, \vec{p}_1)}(1^n) = \alpha] \geq 1 - Q(n, k_n)/|R|.$$

where $\vec{p}_b = \{(m_b[i], S_i)\}_{i=1}^{c(n)}, \{(z[i], T_i)\}_{i=1}^{q(n)}$ and $c(n)$ and $q(n)$ are the lengths of \vec{S}_n and \vec{T}_n respectively.

Note that validity is the special case of strong validity where we restrict to the case when $p(n) = 1$.

Theorem 42. *A message sampler M is strongly $\{(\vec{S}_n, \vec{T}_n)\}_{n \in \mathbb{N}}$ -respecting if and only if it is $\{(\vec{S}_n, \vec{T}_n)\}_{n \in \mathbb{N}}$ -respecting.*

Proof. The "only if" direction is trivial (as mentioned, if $p(n) = 1$ strong validity collapses down to validity). To prove the "if direction", consider some M , $p(\cdot)$, security parameter $n \in \mathbb{N}$, $(\text{sp}, \text{pp}) \in \text{InstGen}(1^n, 1^{k(n)})$ where pp defines a ring R , and oracle machine A (the algebraic adversary) such that $A(1^n)$ makes at most $p(n)$ oracle queries. From semantic security of \mathcal{E} , we have that there exists some polynomial $Q(\cdot, \cdot)$ such that for every (\vec{S}, \vec{T}) -respecting arithmetic circuit C , there exists a constant $c_C \in \{0, 1\}$ such that for every $b \in \{0, 1\}$,

$$\Pr[(\vec{m}_0, \vec{m}_1, \vec{z}) \leftarrow M(1^n, \text{pp}) : \text{isZero}(C(\vec{m}_b, \vec{z})) \neq c] \leq Q(n, k(n))/|R|$$

For $b \in \{0, 1\}$, consider an execution of both $A^{M(\text{pp}, \vec{p}_b)}(1^n)$ where $\vec{m}_0, \vec{m}_1, \vec{z}$ are sampled by M . Note that except with probability $Q(n, k(n))/|R|$ it holds the first oracle query C_1 by A is answered as c_{C_1} . Analogously, if the first i queries C_1, \dots, C_i were answered as c_{C_1}, \dots, c_{C_i} , then except with probability $Q(n, k(n))/|R|$, the $(i+1)$ th query C_{i+1} will be answered as $c_{C_{i+1}}$. It follows that except with probability $p(n)Q(n, k(n))/|R|$ over $\vec{m}_0, \vec{m}_1, \vec{z}$, the output of A is identical to the output of an execution of A where *every* oracle query C is answered by the bit c_C . Thus, for every algebraic attacker A there exists some string α —namely the output of A where every oracle query C is answered by c_C —such that for $b \in \{0, 1\}$, except with probability $p(n)Q(n, k(n))/|R|$, the output of $A^{M(\text{pp}, \vec{p}_b)}(1^n)$ is α . \square

Note that for the above proof to go through it is crucial that we restrict the algebraic attacker to making polynomially-many (or subexponentially-many) oracle queries. This is not just an anomaly of the proof: if we allow the attacker to make an unbounded number of queries, then strong validity would no longer imply validity; we discuss this point further in Section 3.6.2.

3.6.2 Uber Assumptions for Multilinear Encodings

A natural question is whether there are reasonable qualitative strengthenings of semantical security that can be used to achieve stronger notions of obfuscation, such as differing-input (a.k.a. extractability) obfuscation. We here consider such a strengthening.

At first sight, it may seem like the most natural way of defining security of multilinear encodings would be to require that for specific classes of problems, generic attacks cannot be beaten (this is the approach alluded to in [9]). A natu-

ral “uber assumption” (similar to the uber-assumption of [25] in the context of bilinear maps) would be to require that “if an algebraic decisional assumption holds w.r.t. to generic attacks, then it also holds with respect to nuPPT attackers”. Let us now formalize this notion.

Extractable Uber Security

We start by defining a notion of a *computationally valid* message sampler: roughly speaking, we want to capture the intuition that no generic attacker can distinguish \vec{m}_0, \vec{z} from \vec{m}_1, \vec{z} . To get a definition that is as strong as possible, we require indistinguishability to hold in a *pointwise* sense: with overwhelming probability, the output of $A^{\mathcal{M}(\text{pp}, \vec{p}_0)}(1^n, \text{pp})$ is required to be the same as the output of $A^{\mathcal{M}(\text{pp}, \vec{p}_1)}(1^n, \text{pp})$.

Definition 48 (Computationally Respecting Message Sampler). *We say that a nuPPT M is a computationally $\{(\vec{S}_n, \vec{T}_n)\}_{n \in \mathbb{N}}$ -respecting message sampler (or computationally valid w.r.t. $\{(\vec{S}_n, \vec{T}_n)\}_{n \in \mathbb{N}}$) if it satisfies the same conditions as in Definition 34 but where the second bullet is replaced by the following:*

- For every nuPPT oracle machine A , there exists some negligible function ε such that for every $n \in \mathbb{N}$,

$$\Pr[(\text{sp}, \text{pp}) \leftarrow \text{InstGen}(1^n, 1^{k_n}), (\vec{m}_0, \vec{m}_1, \vec{z}) \leftarrow M(1^n, \text{pp}) :$$

$$A^{\mathcal{M}(\text{pp}, \vec{p}_0)}(1^n, \text{pp}) \neq A^{\mathcal{M}(\text{pp}, \vec{p}_1)}(1^n, \text{pp})] \leq \varepsilon(n)$$

where $\vec{p}_b = \{(m_b[i], S_i)\}_{i=1}^{c(n)}, \{(z[i], T_i)\}_{i=1}^{q(n)}$ and $c(n)$ and $q(n)$ are the lengths of \vec{S}_n and \vec{T}_n respectively.

Note that computational validity differs from strong validity (which is equivalent to “plain” validity) in two main aspects: 1) we no longer require the

output of the algebraic attacker to be *constant* with overwhelming probability; rather, we only require that it cannot tell apart \vec{m}_0 and \vec{m}_1 , and 2) the algebraic attacker is restricted to be nuPPT (as opposed to being unbounded and only making polynomially many queries).

We now define *extractable “uber security”* in exactly the same way as semantic security except that we only require the message sampler to be computationally valid (and define entropic uber security in the analogous way). In other words, extractable uber security implies that whenever \vec{m}_0, \vec{z} and \vec{m}_1, \vec{z} are *point-wise computationally indistinguishable* w.r.t. legal algebraic attackers, encodings of them computationally indistinguishable. (We use the term “extractable” since this notion of security requires that if encodings can be distinguished, then we can efficiently find (or “extract”) set-respecting circuits that distinguish the elements.)

We now have the following theorem.

Theorem 43. *Assume the existence of a leveled Fully Homomorphic Encryption scheme with decryption in NC^1 . Then no graded encoding scheme satisfies entropic extractable uber security.*

Proof. Consider any graded encoding scheme \mathcal{E} . To show that \mathcal{E} is not entropic extractable uber secure we need to show that there exists an entropic computationally respecting message sampler M and PPT adversary A such that A distinguishes between encodings of (\vec{m}_0, \vec{z}) and (\vec{m}_1, \vec{z}) where $(\vec{m}_0, \vec{m}_1, \vec{z}) \leftarrow M$.

Our M will sample obfuscations of the following circuit family, that was shown to be unobfuscatable in the virtual black box setting [12]. Let $(\text{Gen}, \text{Enc}, \text{Dec}, \text{Eval})$ be a semantically secure fully homomorphic encryption

scheme with ciphertext size $N(\cdot)$; for simplicity of exposition, let us first assume that it is an “unleveled” FHE. For each security parameter n , consider the class of circuits

$$\mathcal{C}_n = \{C_{n,a,b,v,\text{pk},\text{sk},\hat{a}}\}_{a,b \in \{0,1\}^n, v \in \{0,1\}, (\text{pk},\text{sk}) \in \text{Gen}(1^n), \hat{a} \in \text{Enc}(\text{pk},a)}$$

taking $N(n)$ -bit inputs, where

$$C_{n,a,b,v,\text{pk},\text{sk},\hat{a}}(x) = \begin{cases} (\text{pk}, \hat{a}) & \text{if } x = 0 \\ b & \text{if } x = a \\ v & \text{if } \text{Dec}(\text{sk}, x) = b \\ 0 & \text{otherwise} \end{cases}$$

Then $M(1^n, \text{pp})$ operates as follows, given public parameters pp to a graded encoding scheme it first computes the ring $R = \mathbb{Z}_p$ associated with pp .

- M samples $(\text{pk}, \text{sk}) \leftarrow \text{Gen}(1^n)$ and $a, b \leftarrow \{0, 1\}^n$ uniformly at random, and computes $\hat{a} = \text{Enc}(\text{pk}, a)$.
- M generates branching programs BP_0 and BP_1 corresponding to $C_{n,a,b,0,\text{pk},\text{sk},\hat{a}}$ and $C_{n,a,b,1,\text{pk},\text{sk},\hat{a}}$ respectively, and computes $\widehat{BP}_0 = \text{Merge}(BP_0, BP_1, 0)$ and $\widehat{BP}_1 = \text{Merge}(BP_0, BP_1, 1)$, each of width 10 and length m . Recall, from Claim 24, that \widehat{BP}_0 and \widehat{BP}_1 differ only in levels 1 and m , and that \widehat{BP}_0 and \widehat{BP}_1 are functionally equivalent to BP_0 and BP_1 respectively.
- M samples m random invertible matrices over $\mathbb{Z}_p^{10 \times 10}$, $\{R_i\}_{i \in [m]}$ and $2m$ random scalars from \mathbb{Z}_p , $\{\alpha_{i,b}\}_{i \in [m], b \in \{0,1\}}$. M then uses these matrices and scalars to randomize \widehat{BP}_0 and \widehat{BP}_1 as described by $\text{Rand}(\cdot, p)$ to obtain $\{\alpha_{i,b} \cdot \tilde{B}_{i,b}\}_{i \in [m], b \in \{0,1\}}$, $\{\alpha_{i,b} \cdot \tilde{B}'_{i,b}\}_{i \in [m], b \in \{0,1\}}$ and \mathbf{t} .

- M outputs

$$\vec{m}_0 = (\{\alpha_{1,b} \cdot \tilde{B}_{1,b}\}_{b \in \{0,1\}}, \{\alpha_{m,b} \cdot \tilde{B}_{m,b}\}_{b \in \{0,1\}})$$

$$\vec{m}_1 = (\{\alpha_{1,b} \cdot \tilde{B}'_{1,b}\}_{b \in \{0,1\}}, \{\alpha_{m,b} \cdot \tilde{B}'_{m,b}\}_{b \in \{0,1\}})$$

$$\vec{z} = (\{\alpha_{i,b} \cdot \tilde{B}_{i,b}\}_{i \in [m']/\{1,m\}, b \in \{0,1\}}, \mathbf{t})$$

Note that (\vec{m}_0, \vec{z}) is identically distributed to $\text{Rand}(\widehat{BP}_0, p)$ and similarly (\vec{m}_1, \vec{z}) is identically distributed to $\text{Rand}(\widehat{BP}_1, p)$. As a result, by Proposition 2, we have that M is an entropic message sampler.

Let $(\{S_{i,b}\}_{i \in [m], b \in \{0,1\}}, S_{\mathbf{t}}) = \text{SetSystem}(m, N, \text{inp})$, where inp is the labelling function for the branching programs \widehat{BP}_0 and \widehat{BP}_1 , and let

$$\vec{S}_n = \{S_{1,b}, S_{m,b}\}_{b \in \{0,1\}}$$

$$\vec{T}_n = (\{S_{i,b}\}_{i \in [m']/\{1,m\}, b \in \{0,1\}}, S_{\mathbf{t}})$$

We show that M is a computationally $\{\vec{S}_n, \vec{T}_n\}_{n \in \mathcal{N}}$ -respecting message sampler, *i.e.* no nuPPT oracle machine A' can pointwise distinguish the oracles $\mathcal{M}(\vec{m}_0, \vec{z})$ and $\mathcal{M}(\vec{m}_1, \vec{z})$. We note that by Lemma 32 and a Union Bound over A' 's queries, the output of $A'^{\mathcal{M}(\vec{m}_0, \vec{z})}$ (resp. $A'^{\mathcal{M}(\vec{m}_1, \vec{z})}$) can be simulated with only oracle access to BP_0 (resp. BP_1), or equivalently, to $C_{n,a,b,0,\text{pk},\text{sk},\hat{a}}$ (resp. $C_{n,a,b,1,\text{pk},\text{sk},\hat{a}}$)²⁹. In fact, with high probability over the randomness of M , A' and the simulator, the simulator's output is identical to the output of A' . We observe that this simulation can be made *efficient* using the techniques introduced in [9] (*i.e.* by modifying BP_0 and BP_1 to be *dual-input* branching programs and correspondingly changing SetSystem); this requires encodings elements using sets of size 4 (as opposed to 2 as in our original construction). Let this efficient simulator be Sim .

²⁹To apply the Union Bound it is important that the query response $C(\vec{m}_b, \vec{z})$ depends only on the queried arithmetic circuit C and the input-output behavior of BP_b as shown in Lemma 32

We would now like to argue that with high probability over the randomness of M and Sim , $\text{Sim}^{BP_0} = \text{Sim}^{BP_1}$. Recall that the circuits $C_{n,a,b,0,\text{pk},\text{sk},\hat{a}}$ (equivalent to BP_0) and $C_{n,a,b,1,\text{pk},\text{sk},\hat{a}}$ (equivalent to BP_1) differ only on inputs x for which $\text{Dec}(\text{sk}, x) = b$ (on these inputs $C_{n,a,b,0,\text{pk},\text{sk},\hat{a}}(x) = 0$, whereas $C_{n,a,b,1,\text{pk},\text{sk},\hat{a}}(x) = 1$). Since b was randomly chosen from an exponentially large set of values, to find such an input with noticeable probability, Sim must query one of the circuits on input a with noticeable probability, otherwise its view is independent of b . However, if the original ciphertext \hat{a} is an encryption of 0 instead of a , then the view of Sim is independent of a , and thus Sim can only query a with negligible probability. Thus by the semantic security of the FHE scheme, the probability that Sim can query a when given BP_0 or BP_1 is negligible. This implies that the outputs of Sim^{BP_0} and Sim^{BP_1} differ with only negligible probability.

We now have that :

- $A^{\mathcal{M}(\vec{m}_0, \vec{z})} = \text{Sim}^{BP_0}$, except with negligible probability;
- $\text{Sim}^{BP_0} = \text{Sim}^{BP_1}$, except with negligible probability;
- $\text{Sim}^{BP_1} = A^{\mathcal{M}(\vec{m}_1, \vec{z})}$, except with negligible probability.

By a union bound, we have that $A^{\mathcal{M}(\vec{m}_0, \vec{z})} = A^{\mathcal{M}(\vec{m}_1, \vec{z})}$, except with negligible probability. Thus M must be a computationally respecting sampler. Finally, it follows using identically the same argument as in Section 3.4.3 that the message sampler satisfies the required high-entropy condition and thus is an entropic computationally respecting message sampler.

Now we will show an $nuPPT$ adversary A that distinguishes between encodings of (\vec{m}_0, \vec{z}) and (\vec{m}_1, \vec{z}) when encoded under sets (\vec{S}_n, \vec{T}_n) . Note that given en-

codings of one of (\vec{m}_0, \vec{z}) and (\vec{m}_1, \vec{z}) , A in fact receives either $\text{Obf}(\widehat{BP}_0)$ or $\text{Obf}(\widehat{BP}_1)$. Let us refer to this input to A as O .

A evaluates O on input 0 to receive (pk, \hat{a}) , and then simply homomorphically evaluates O on the ciphertext \hat{a} in order to generate a valid encryption of the hidden value b , and then feeds this new ciphertext back into O to reveal the secret bit v , and then outputs v . Thus A succeeds in distinguishing (\vec{m}_0, \vec{z}) and (\vec{m}_1, \vec{z}) with probability 1. Additionally, note that since O is a constant-width branching program, O can be computed by a NC^1 circuit, thus for this argument it suffices to use a leveled FHE.

We thus have that no graded encoding scheme can satisfy entropic extractable uber security. \square

“Plain” Uber Security

Due to the above impossibility result, we here consider a weaker variant of an uber security—which we simply refer to as (plain) “uber security”, where we strengthen the “computational validity” condition to a “weak validity” condition where the algebraic attacker is allowed to be unbounded while making polynomially many queries. Note that weak validity differs from strong validity only in the respect that weak validity does not require the output of the algebraic attacker is *constant* (with overwhelming probability).

Definition 49 (Weakly Respecting Message Sampler). *We say that a nuPPT M is a weakly $\{(\vec{S}_n, \vec{T}_n)\}_{n \in \mathbb{N}}$ -respecting message sampler (or weakly valid w.r.t. $\{(\vec{S}_n, \vec{T}_n)\}_{n \in \mathbb{N}}$) if it satisfies the same conditions as in Definition 34 but where the second bullet is replaced by the following:*

- For every polynomial p , there exists some polynomial Q such that for every $n \in \mathbb{N}$, every (sp, pp) in the support of $\text{InstGen}(1^n, 1^{k_n})$, every (deterministic) oracle algorithm A that on input 1^n makes at most $p(n)$ oracle queries,

$$\Pr[(\vec{m}_0, \vec{m}_1, \vec{z}) \leftarrow M(1^n, \text{pp}) : A^{\mathcal{M}(\text{pp}, \vec{p}_0)}(1^n) = A^{\mathcal{M}(\text{pp}, \vec{p}_1)}(1^n)] \geq 1 - Q(n, k_n)/|R|.$$

where $\vec{p}_b = \{(m_b[i], S_i)\}_{i=1}^{c(n)}, \{(z[i], T_i)\}_{i=1}^{q(n)}$ and $c(n)$ and $q(n)$ are the lengths of \vec{S}_n and \vec{T}_n respectively.

We define “uber security” in exactly the same way as semantic security except that we only require the message sampler to be weakly valid (and define entropic uber security in the analogous way). In other words, uber security implies that whenever \vec{m}_0, \vec{z} and \vec{m}_1, \vec{z} are *pointwise statistically close* w.r.t. legal algebraic attackers, encodings of them computationally indistinguishable.

Let us remark that for uber security to imply semantical security, it is important that we restrict the algebraic attacker (in the definition of a weakly valid message sampler) to only make polynomially many queries. Otherwise, even the aGDDH distribution (described in Section 3.3) is not weakly valid: With high probability over (m_0, m_1, \vec{z}) sampled from the aGDDH distribution, there always exists *some* legal arithmetic circuit C such that $\text{isZero}(C(m_0, \vec{z})) \neq \text{isZero}(C(m_1, \vec{z}))$.³⁰ Therefore, an unbounded-query algebraic adversary could simply go over all legal arithmetic circuits and distinguish the elements.

We are not aware of any attacks (like those against *extractable* uber security) against “plain” uber security, and it thus seems like a reasonable strengthening of semantical security, which may have other applications. In fact, we may

³⁰Consider a very simple aGDDH instance, where $|\vec{z}| = 2, T_1 = T_2 = S = [k]$. For non-zero z_1, z_2 , there always exists some a such that the circuit $C(m, z_1, z_2) = \text{isZero}(m - az_1)$ yields different outputs on input (m_0, \vec{z}) and (m_1, \vec{z}) —namely, $a = z_2$.

consider an even further strengthening of this notion—which we refer to as *statistical uber security*—by replacing the the weakly valid message sampler by a *super weakly valid* message sampler which only requires \vec{m}_0, \vec{z} and \vec{m}_1, \vec{z} to be *statistically indistinguishable* by algebraic attackers (as opposed to be *pointwise* statistically indistinguishable); that is, the second bullet in Definition 34 is replaced by:

- For every (computationally unbounded) oracle machine A that makes at most polynomially many oracle queries, there exists a negligible function ε such that for every security parameter $n \in \mathbb{N}$,

$$\begin{aligned} & |Pr[(sp, pp) \leftarrow \text{InstGen}(1^n, 1^{k(n)}), (\vec{m}_0, \vec{m}_1, \vec{z}) \leftarrow M(1^n, pp) : \\ & \quad A^{\mathcal{M}(pp, \vec{p}_0)}(1^n, pp) = 1] - \\ & Pr[(sp, pp) \leftarrow \text{InstGen}(1^n, 1^{k(n)}), (\vec{m}_0, \vec{m}_1, \vec{z}) \leftarrow M(1^n, pp) : \\ & \quad A^{\mathcal{M}(pp, \vec{p}_1)}(1^n, pp) = 1]| \leq \varepsilon(n) \end{aligned}$$

where $\vec{p}_b = \{(m_b[i], S_i)\}_{i=1}^{c(n)}, \{(z[i], T_i)\}_{i=1}^{q(n)}$ and $c(n)$ and $q(n)$ are the lengths of \vec{S}_n and \vec{T}_n respectively.

3.6.3 Strong Semantical and Uber Security

Recall that in the definition of both validity and weak validity, we consider *arbitrary-size* set-respecting circuits. We may weaken both validity conditions (and thus obtain stronger notion of semantical and uber security) by restricting attention to only polynomial-size arithmetic circuits. Note that in the context of uber security, this takes us a step closer to extractable uber security (which is impossible under reasonable assumption): we restrict to algebraic attackers

that make polynomially-many queries and each query is polynomial-size, but the attacker may generate these queries (and generate its final output) in a computationally unbounded way. We refer to these notions respectively as *strong semantical security* and *strong uber security*.

3.6.4 Weak Semantic Security

We end this section by considering a *weaker* notion of semantical security—let us refer to it as *weak semantical security*—where the definition of a valid message sampler requires the the answer to every set-respecting circuit is *actually* constant (as opposed to only being constant with overwhelming probability); a similar relaxation can be applied also to uber security. While we do not know whether any of these weaker assumptions suffices for obtaining obfuscation (and they do not imply the aGDDH assumption), the weak notion of semantical security suffices for obtaining *witness encryption* [59]—roughly speaking, the notion of witness encryption enables a sender to encrypt a message m using an NP-statement x such that a) if the statement is false, then encodings of any two messages are indistinguishable, and b) if the statement is true, then anyone who has a witness w for x can recover m . Let us briefly sketch this construction.³¹ As in [59], we focus on the NP-language **Exact-Cover** where an x instance consist of sets $S_1, \dots, S_n \subseteq [k]$; for a true instance, there exists some “exact cover” of $[k]$ using a subset of the sets, whereas for a false instance no such exact cover exists. Now, to encrypt the bit m under the instance S_1, \dots, S_n , use a multilinear encoding scheme over the set $[k + 1]$, encode 1 under each of the sets S_1, \dots, S_n and finally encode m under the set $\{k + 1\}$. Clearly anyone who knows an exact

³¹The observation that semantically secure multilinear encoding directly implies witness encryption was obtained in a conversation with Sanjam Garg, Craig Gentry and Shai Halevi.

cover can obtain an encoding of m under $[k + 1]$ (by appropriately multiplying the sets corresponding to the exact cover and additionally the encoding of m under $\{k + 1\}$). On the other hand, if the instance is false, there is no exact cover, and thus “legal” algebraic operation can never be used to obtain an encoding under the full set $[k + 1]$ and thus zero-testing can never be used; thus indistinguishability of encryptions follows by weak semantical security.

3.7 Technical Lemma

Claim 44. Fix $m, w \in \mathcal{N}$, and let $p \in \mathcal{N}$ be a prime. Let \mathcal{D}_0 be the following distribution:

$$\mathcal{D}_0 = \{\{R_i\}_{i \in [m]}, \{\alpha_{i,b}\}_{i \in [m], b \in \{0,1\}}\}$$

where each R_i is a uniformly random invertible matrix in $\mathbb{Z}_p^{w \times w}$ (i.e. $\det(R_i) \neq 0$, and each $\alpha_{i,b}$ is a uniformly random non-zero scalar in \mathbb{Z}_p .

Let \mathcal{D}_1 be a distribution defined identically to \mathcal{D}_0 , except with each R_i being a uniformly random (not necessarily invertible) matrix in $\mathbb{Z}_p^{w \times w}$, and each $\alpha_{i,b}$ a uniformly random (not necessarily non-zero) scalar in \mathbb{Z}_p .

Then:

$$\Delta(\mathcal{D}_0, \mathcal{D}_1) \leq 8wm/p$$

where $\Delta(\mathcal{D}_0, \mathcal{D}_1)$ denotes the statistical distance between distributions \mathcal{D}_0 and \mathcal{D}_1 .

Proof. Note that \mathcal{D}_0 and \mathcal{D}_1 are each uniformly distributed on their respective supports, and that $\text{supp}(\mathcal{D}_0) \subseteq \text{supp}(\mathcal{D}_1)$. Then the statistical distance between

\mathcal{D}_0 and \mathcal{D}_1 can be computed as follows:

$$\begin{aligned}
\Delta(\mathcal{D}_0, \mathcal{D}_1) &= \sum_{d \in \text{supp}(\mathcal{D}_0) \cup \text{supp}(\mathcal{D}_1)} |\Pr[\mathcal{D}_0 = d] - \Pr[\mathcal{D}_1 = d]| \\
&= \sum_{d \in \text{supp}(\mathcal{D}_0)} |\Pr[\mathcal{D}_0 = d] - \Pr[\mathcal{D}_1 = d]| + \sum_{d \in \text{supp}(\mathcal{D}_1) \setminus \text{supp}(\mathcal{D}_0)} |\Pr[\mathcal{D}_1 = d]| \\
&= \sum_{d \in \text{supp}(\mathcal{D}_0)} \left| \frac{1}{|\text{supp}(\mathcal{D}_0)|} - \frac{1}{|\text{supp}(\mathcal{D}_1)|} \right| + \sum_{d \in \text{supp}(\mathcal{D}_1) \setminus \text{supp}(\mathcal{D}_0)} \left| \frac{1}{|\text{supp}(\mathcal{D}_1)|} \right| \\
&= (|\text{supp}(\mathcal{D}_0)| \cdot \left| \frac{1}{|\text{supp}(\mathcal{D}_0)|} - \frac{1}{|\text{supp}(\mathcal{D}_1)|} \right| + \\
&\quad (|\text{supp}(\mathcal{D}_1) \setminus \text{supp}(\mathcal{D}_0)| \cdot \left| \frac{1}{|\text{supp}(\mathcal{D}_1)|} \right|) \\
&= 2 \cdot \left(1 - \frac{|\text{supp}(\mathcal{D}_0)|}{|\text{supp}(\mathcal{D}_1)|} \right)
\end{aligned}$$

But notice that $(1 - \frac{|\text{supp}(\mathcal{D}_0)|}{|\text{supp}(\mathcal{D}_1)|})$ can be interpreted as $\Pr[\exists i \in [m], b \in \{0, 1\} : \det(R_i) = 0 \vee \alpha_{i,b} = 0]$. For each $i \in [m]$, the probability $\det(R_i) = 0$ can be bounded by applying the Schwartz-Zippel lemma to the $\det(\cdot)$, which is a polynomial of degree w . Thus we have that $\Pr[\det(R_i) = 0] \leq w/p$. Further, each $\alpha_{i,b}$ is zero with probability $1/p$. Hence, applying a union bound, we have that

$$\begin{aligned}
\Delta(\mathcal{D}_0, \mathcal{D}_1) &= 2 \cdot \left(1 - \frac{|\text{supp}(\mathcal{D}_0)|}{|\text{supp}(\mathcal{D}_1)|} \right) \\
&\leq 2 \cdot (2m/p + mw/p) \\
&\leq 8wm/p
\end{aligned}$$

□

3.8 Proof of Lemma 33

In this section, we prove Lemma 33, restated below for clarity:

Lemma 37. Fix $m, n, w \in \mathcal{N}$ and $\text{inp} : [m] \rightarrow [n]$. Let $\vec{S} = \text{SetSystem}(m, n, \text{inp}) = (\{S_{i,b}\}_{i \in [m], b \in \{0,1\}}, S_t)$, and let C be any weakly \vec{S} -respecting arithmetic circuit whose out-

put wire is tagged with $T \subseteq [k]$. Then there exists a set $U \subseteq \{0, 1, *\}^m$ such that for every branching program BP of width w and length m on n input bits, with input tagging function inp , every prime p , and every $(\{\alpha_{i,b} \cdot \tilde{B}_{i,b}\}_{i \in m, b \in \{0,1\}}, \mathbf{t}) \leftarrow \text{Rand}(BP, p)$,

(i)

$$C(\{\alpha_{i,b} \tilde{B}_{i,b}\}_{i \in [m], b \in \{0,1\}}, \mathbf{t}) \equiv \sum_{u \in U} C_u(\{\alpha_{i,b} \tilde{B}_{i,b}\}_{i \in [m], b \in \{0,1\}}, \mathbf{t})$$

where each C_u is a weakly \vec{S} -respecting arithmetic circuit, whose input wires are tagged only with sets $\in \{S_{i,u[i]}\}_{i \in [m]: u[i] \neq *} \cup \{S_t\}$, and whose output wire is tagged with T .

(ii) Each C_u above is the sum of several “monomial” circuits, where each monomial circuit performs only multiplications of elements in $(\{\alpha_{i,b} \cdot \tilde{B}_{i,b}\}_{i \in m, b \in \{0,1\}}, \mathbf{t})$, is weakly \vec{S} -respecting, and has output wire tagged with T .

(iii) For each C_u above,

$$C_u(\{\alpha_{i,b} \tilde{B}_{i,b}\}_{i \in [m], b \in \{0,1\}}, \mathbf{t}) = \alpha_u \cdot p_u(\{\tilde{B}_{i,u[i]}\}_{i \in [m]: u[i] \neq *}, \mathbf{t})$$

where p_u is some polynomial, and $\alpha_u = (\prod_{i \in [m]: u[i] \neq *} \alpha_{i,u[i]})$. Furthermore, when p_u is viewed as a sum of monomials, each monomial contains exactly one entry from each $\tilde{B}_{i,u[i]}$ such that $u[i] \neq *$, and possibly one entry from \mathbf{t} . Further, p_u can be computed by a weakly \vec{S} -respecting circuit whose output wire is tagged with T .

Proof. Part (i) We begin by expressing the circuit C as a polynomial in variables $(\{\alpha_{i,b} \cdot \tilde{B}_{i,b}\}_{i \in m, b \in \{0,1\}}, \mathbf{t})$, in the form of a sum of monomials (possibly exponentially many). We do so recursively: we associate each wire w of the circuit with a multiset S_w of pairs of monomials and signs (“+1” or “-1”), such that the sum of the monomials multiplied by their respective signs computes the same value as the value computed by the circuit at that wire. We eventually output the

multiset of monomial pairs corresponding to the output wire. We compute the sets of monomials as follows:

- Any input wire of the circuit reading input variable v can be represented as the set $\{(v, +)\}$.
- The output wire of an addition gate can be represented as the union of the multisets of monomial pairs representing the gates left and right children.
- The output wire of an subtraction gate can be similarly represented as the union of the multisets of the gate's left input wire, and of its right input wire with the "sign" component of every pair negated (from "+1" to "-1" and vice versa), to reflect subtraction.
- For the output wire of a multiplication gate, for each pair (M_1, s_1) in the multiset of its left input and each pair (M_2, s_2) in the multiset of its right input, we add $(M_1 \cdot M_2, s_1 \cdot s_2)$ to the multiset of the output wire.

We note that it holds inductively in the above process that the sum of the monomials in the multiset associated with each wire w in C , multiplied by its appropriate sign, equals the value computed on that wire w .

We also show that each monomial in the set corresponding to a wire can be computed by a weakly \vec{S} -respecting circuit whose output wire has the same tag as the wire. This can again be seen inductively:

- This property holds at any input wire of C , since the only monomial in the set can be computed using the input wire itself as the "monomial circuit".
- This property also holds at any output wire of an addition or subtraction gate, since the circuit corresponding to any monomial in this wire's set is

the same as the circuit for the monomial from the corresponding incoming wire to the gate.

- Finally, at the output wire of a multiplication gate G , for any monomial M in this wire's set computed as the product of monomials $M1$ and $M2$, the circuit for M is simply the circuit for each of $M1$ and $M2$, joined by a multiplication gate. Since G performs a set respecting multiplication, and the output wires of $M1$ and $M2$'s circuits have the same tags as the input wires of G , we have that the multiplication joining $M1$ and $M2$'s circuits to produce M 's circuit is set-respecting, and so the circuit corresponding to M is a weakly \vec{S} -respecting circuit whose output wire has the same tag as the output wire of G .

Thus each of the monomials in the decomposition of C can be represented as a weakly set-respecting arithmetic circuit with output wire tagged with T , where this circuit simply multiplies together all terms in the monomial in some order, and performs no additions. Finally, the tags of the input wires of these monomial circuits must be mutually disjoint, otherwise the monomial circuit would perform a non-set-respecting multiplication at some level.

We label each monomial M with an element $u \in \{0, 1, *\}^m$, where $u[i] = b$ if $S_{i,b}$ is the label on one of input wires in M 's circuit representation, and $u[i] = *$ if neither $S_{i,0}$ and $S_{i,1}$ are labels on any of M 's input wires. We note that no monomial can have both $S_{i,0}$ and $S_{i,1}$ on its input wires because these two sets are not disjoint, and the tags of the input wires of the monomial circuits must be mutually disjoint.

We now let C_u be the circuit representing the subtraction of all monomials in the the decomposition of C labelled with u and sign (-1) from the sum of all

monomials in the the decomposition of C labelled with u and sign $(+1)$. Since each monomial can be represented as a weakly set-respecting circuit with output wire tagged with T , adding several monomials together is a set-respecting operation, as is subtracting several monomials from the sum, and thus each C_u is a weakly set-respecting circuit. Further, since each monomial circuit has output wire tagged with T , each C_u also has output wire tagged with T . Further, by the way we labelled each monomial, each of the input wires of C_u is tagged only with sets $\in \{S_{i,u[i]}\}_{i \in [m]:u[i] \neq * } \cup \{S_t\}$. Finally, if we sum over all the u , we capture all the monomials in the decomposition of C multiplied by their respective signs, so we have that $\sum_u C_u = C$.

Part (ii) We observe that by construction of C_u , it is a sum of several monomial circuits each of which performs only multiplications of its inputs, is weakly \vec{S} -respecting, and has output wire tagged with T .

Part (iii) From part (ii), we have that for each C_u , it is a sum of several monomial circuits each of which performs only multiplications of its inputs, is weakly \vec{S} -respecting, and has output wire tagged with T . Furthermore, for each such monomial circuit the input tags are drawn from sets $\in \{S_{i,u[i]}\}_{i \in [m]:u[i] \neq * } \cup \{S_t\}$. In fact, each of these monomials must contain exactly one input wire tagged with each of the sets in $\{S_{i,u[i]}\}_{i \in [m]:u[i] \neq * }$ and exactly one set tagged with S_t if and only if $S_t \subseteq T$. This means that each of these monomials is the product of one element chosen from each of the matrices $(\{\alpha_{i,u[i]} \cdot \tilde{B}_{i,u[i]}\}_{i \in [m]:u[i] \neq * }$ and possibly one element from \mathbf{t} . Thus each monomial in the decomposition of C_u has a common factor of $\alpha_u = (\prod_{i \in [m]:u[i] \neq * } \alpha_{i,u[i]})$.

We can now write C_u as a polynomial (namely the sum of its monomials multiplied by their respective signs), and by factoring α_u from each of it monomials

and letting p_u be the remaining polynomial, we have, as required, that

$$C_u(\{\alpha_{i,b}\tilde{B}_{i,b}\}_{i \in [m], b \in \{0,1\}}, \mathbf{t}) = \alpha_u \cdot p_u(\{\tilde{B}_{i,u[i]}\}_{i \in [m]:u[i] \neq *}, \mathbf{t})$$

Finally, we note that computing p_u is the same as computing C_u if the alphas are set to 1. Since C_u is \vec{S} -respecting, we thus have that p_u can be computed by a weakly \vec{S} -respecting circuit whose output wire is tagged with T . \square

3.9 Proof of Lemma 21

In this section we prove Lemma 21, restated below for clarity.

Lemma 45. *Let $c, \varepsilon \in \mathbb{N}$ and \mathcal{E} be an (c, k^ε) -semantically secure encoding scheme. Then for every polynomial $q(k)$ there exists a $(c, q(k))$ -semantically secure encoding scheme.*

Proof. Consider any polynomial $q(\cdot)$ and constants c, ε . Given a (c, k^ε) -semantically secure encoding \mathcal{E} , we construct a new multilinear encoding scheme \mathcal{E}' and prove that \mathcal{E}' is $(c, q(k))$ -semantically secure. Let $(\text{InstGen}, \text{Enc}, \text{Add}, \text{Sub}, \text{Mult}, \text{isZero})$ be the algorithms associated with \mathcal{E} . We define a new encoding scheme $\mathcal{E}' = (\text{InstGen}', \text{Enc}', \text{Add}', \text{Sub}', \text{Mult}', \text{isZero}')$ as follows.

- $\text{InstGen}'$ on input $(1^n, 1^k)$ runs $(\text{pp}, \text{sp}) \leftarrow \text{InstGen}(1^n, 1^{(q(k)+1)^{1/\varepsilon}})$ and generates an encoding of a uniformly random non-zero element e under the set $\{k+1, \dots, (q(k)+1)^{1/\varepsilon}\}$ by running $u^1 \leftarrow \text{Enc}(\text{sp}, e, \{k+1, \dots, (q(k)+1)^{1/\varepsilon}\})$. $\text{InstGen}'$ outputs (pp, u^1) as the public parameters and sp as the secret parameters.
- $\text{Enc}', \text{Add}', \text{Sub}', \text{Mult}'$ are identical to $\text{Enc}, \text{Add}, \text{Sub}, \text{Mult}$ respectively.

- isZero' takes as input public parameters (pp, u^1) and an encoding u under the set $[k]$ to zero-test. isZero' simply outputs $\text{isZero}(\text{Mult}(\text{pp}, u, u^1))$. The correctness of isZero' follows from that of isZero and the fact that $\text{Mult}(\text{pp}, u, u^1)$ returns an encoding, under the set $[(q(k) + 1)^{1/\varepsilon}]$, of an element which is zero if and only if u is an encoding of zero.

It is easy to see that the correctness of \mathcal{E}' follows from that of \mathcal{E} .

We now show that \mathcal{E}' is $(c, q(k))$ -semantically secure. Assume for contradiction there exists a polynomial $k'(\cdot)$, ensemble $\{\vec{S}'_n, \vec{T}'_n\}_{n \in \mathbb{N}}$ of sets where $|\vec{S}'_n| = c$, $|\vec{T}'_n| = q(k'(n))$, $\{\vec{S}'_n, \vec{T}'_n\}_{n \in \mathbb{N}}$ -respecting message sampler M' and nuPPT adversary A' such that for sufficiently large n , A' distinguishes encodings of elements as described in the semantic security game in Definition 35.

Let $k(\cdot)$ be a polynomial such that $k(n) = (q(k'(n)) + 1)^{1/\varepsilon}$. For every $n \in \mathbb{N}$, let \vec{S}_n, \vec{T}_n be a sequence of sets over $[k(n)]$ where $\vec{S}_n = \vec{S}'_n$ and $\vec{T}_n = (\vec{T}'_n, \{k'(n) + 1, \dots, k(n)\})$. We will construct a $\{\vec{S}_n, \vec{T}_n\}_{n \in \mathbb{N}}$ -respecting message sampler M and nuPPT adversary A such that (M, A) breaks the (c, k^ε) -semantic security of \mathcal{E} .

We define the message sampler M as follows: on input 1^n , $\text{pp} \in \text{InstGen}(1^n, 1^{k(n)})$, M samples $(\vec{m}_0, \vec{m}_1, \vec{z}) \leftarrow M'(1^n, \text{pp})$. and outputs the elements $(\vec{m}_0, \vec{m}_1, (\vec{z}, e))$ where e is a uniformly random non-zero element, *i.e.* M outputs the same elements sampled by M' with an additional element e . Note that M' samples elements based only on the ring associated with the public parameters pp , which in this case, is the same ring associated with $\text{pp}' \in \text{InstGen}'(1^n, 1^{k'(n)})$.

To show that M is $\{\vec{S}_n, \vec{T}_n\}_{n \in \mathbb{N}}$ -respecting, we claim that for any (\vec{S}'_n, \vec{T}'_n) -respecting circuit C acting on $(\vec{m}_0, \vec{m}_1, (\vec{z}, e))$ there exists a (\vec{S}'_n, \vec{T}'_n) -respecting circuit C' acting on $(\vec{m}_0, \vec{m}_1, \vec{z})$ such that $\text{isZero}(C(\cdot)) = \text{isZero}(C'(\cdot))$. C' is simply the

circuit C computes to obtain an element corresponding to the set $[k'(n)]$, with which it must multiply an element under the set $\{k'(n) + 1, \dots, k(n)\}$ to reach the target set $[k(n)]$. Since M' is $\{\vec{S}'_n, \vec{T}'_n\}_{n \in \mathcal{N}}$ -respecting, the output of $\text{isZero}(C'(\cdot))$ is constant with overwhelming probability. Therefore, the output of $\text{isZero}(C(\cdot))$ is constant with overwhelming probability too, and M is $\{\vec{S}_n, \vec{T}_n\}_{n \in \mathcal{N}}$ -respecting.

We now define a nuPPT adversary A that breaks the semantic security of \mathcal{E} . On input encodings \vec{u} and public parameters pp , A simply removes the last encoding u from \vec{u} and runs A' on input public parameters (pp, u) and the remaining encodings. Observe that for any security parameter n , the output of A in the semantic security game in Definition 35 when played with message sampler M and sets \vec{S}_n, \vec{T}_n is identical to the output of A' in the game played with message sampler M' and sets \vec{S}'_n, \vec{T}'_n . Recall that \vec{S}_n, \vec{T}_n are sequences of sets over $[k(n)]$ and $|\vec{S}_n| = c$ and $|\vec{T}_n| = k(n)^\varepsilon$. Therefore, this contradicts the (c, k^ε) -semantic security of \mathcal{E} . \square

3.10 Acknowledgments

We are very grateful to Benny Applebaum, Omer Paneth, Ran Canetti, Kai-Min Chung, Sanjam Garg, Craig Gentry, Shai Halevi, Amit Sahai, abhi shelat, Hoeteck Wee and Daniel Wichs for many helpful comments. We are especially gratefeul to Shai for pointing out the connection between semantical security for multilinear encodings and the “uber” assumption for bilinear maps of [25], and for several very useful conversations about multilinear encodings and the security of the [54] constructions, to Amit for several helpful conversations about the presentation of our results, and Benny for suggesting we make our proof more

modular (which lead to the notion of neighboring-matrix branching programs).
Finally thanks to the anonymous Crypto reviewers for their useful comments.
Thanks so very much!

CHAPTER 4
OUTPUT-COMPRESSING RANDOMIZED ENCODINGS AND
APPLICATIONS

This chapter contains joint work with Huijia Lin (UCSB), Rafael Pass (Cornell University) and Sidharth Telang (Cornell University), to appear in the proceedings of the Theory of Cryptography Conference (TCC) 2016.

4.1 Introduction

The beautiful notion of a *randomized encoding (RE)*, introduced by Ishai and Kushilevitz [82], aims to trade the computation of a “complex” (deterministic) function Π on a given input x for the computation of a “simpler” randomized function—the “encoding algorithm”—whose output distribution $\hat{\Pi}(x)$ encodes $\Pi(x)$ (from which $\Pi(x)$ can be efficiently decoded, or “evaluated”). Furthermore, the encoding $\hat{\Pi}(x)$ should not reveal anything beyond $\Pi(x)$; this is referred to as the *privacy*, or *security*, property of randomized encodings and is typically defined through the simulation paradigm [76].

Most previous work have focused on randomized encodings where encodings can be computed in lower parallel-time complexity than what is required for computing the original function Π . For instance, all log-space computations have *perfectly-secure* randomized encodings in \mathbf{NC}^0 [82, 83, 5], and assuming low-depth pseudo-random generators, this extends to all polynomial-time computations (with computational security) [6, 118]. Such randomized encodings have been shown to have various applications to parallel cryptography, secure computation, verifiable delegation, etc. (see [4] for a survey).

Bitansky, Garg, Lin, Pass and Telang [19] recently initiated a study of *succinct randomized encodings* where we require that the *time* required to compute $\hat{\Pi}(x)$ is smaller than the time required to compute $\Pi(x)$; their study focused on functions Π that have *single-bit* outputs. [19, 43, 90] show that subexponentially-secure indistinguishability obfuscators (*iO*) [12, 57] and one-way functions¹ imply the existence of such succinct randomized encodings for all polynomial-time Turing machines that output just a single bit.

We here further the study of such objects, focusing on functions Π with *long* outputs. Given a description of a Turing machine Π and an input x , we consider two notions of efficiency for randomized encodings $\hat{\Pi}(x)$ of $\Pi(x)$ with running time T .

- *compact RE*: Encoding time (and thus also size of the encodings) is $\text{poly}(|\Pi|, |x|, \log T)$
- *sublinear RE*: Encoding time (and thus also size) is bounded by $\text{poly}(|\Pi|, |x|) \cdot T^{1-\epsilon}$, for some $\epsilon > 0$.

We assume without loss of generality that the randomized encoding $\hat{\Pi}(x)$ of Π, x itself is a program, and that the decoding/evaluation algorithm simply executes $\hat{\Pi}(x)$.

It is easy to see that for such notions of efficiency, the standard simulation-based notion of security is impossible to achieve—roughly speaking, the simulator given just $\Pi(x)$ needs to output a “compressed” version of it, which is impossible if $\Pi(x)$ has high pseudo-Kolmogorov complexity (e.g., if Π is a PRG); we formalize this argument in Theorem 16 in Section 4.7. Consequently, we

¹The one-way function assumption can be weakened to assume just that $\text{NP} \not\subseteq \text{iOBPP}$ [88].

consider weaker indistinguishability-based notions of privacy. One natural indistinguishability based notion of privacy simply requires that encoding $\hat{\Pi}_0(x_0)$ and $\hat{\Pi}_1(x_1)$ are indistinguishable as long as $\Pi_0(x_0) = \Pi_1(x_1)$ and $\text{Time}(\Pi_0(x_0)) = \text{Time}(\Pi_1(x_1))$, where $\text{Time}(\Pi(x))$ is the running-time of $\Pi(x)$; such a notion was recently considered by Ananth and Jain [3]. In this work, we consider a stronger notion which requires indistinguishability of $\hat{\Pi}_0(x_0)$ and $\hat{\Pi}_0(x_1)$ as long as Π_0, x_0 and Π_1, x_1 are sampled from some distributions such that $\Pi_0(x_0), \text{Time}(\Pi_0(x_0))$ and $\Pi_1(x_1), \text{Time}(\Pi_1(x_1))$ are indistinguishable. We refer to this notion as *distributional indistinguishability security*, and note that it easily follows that the standard simulation-based security implies distributional indistinguishability security.

The goal of this paper is to investigate compact and sublinear RE satisfying the above-mentioned distributional indistinguishability notion. For the remainder of the introduction, we refer to randomized encodings satisfying distributional indistinguishability security as simply *RE*. For comparison, we refer to randomized encodings with the weaker (non-distributional) indistinguishability security as *weak RE*.

We note here that [63] previously introduced a very similar notion of distributional indistinguishability in the setting of reusable garbled circuits. In the CRS model, reusable garbled circuits are equivalent to a “secret key” version of randomized encoding, thus the [63] security notion can be seen as very related to ours. Indeed, similar to our approach in this work, [63] also used distributional indistinguishability to achieve reusable garbled circuits with long outputs, and in doing so, circumvent an impossibility result.²

Compact RE v.s. Obfuscation Before turning to describe our results, let us point

²We thank Daniel Wichs for pointing out the connection to [63].

out that RE can be viewed as (a degenerate form) of obfuscation for special classes of programs.

Recall that an indistinguishability obfuscator (*iO*) [12, 57] is a method \mathcal{O} for “scrambling” a program Π into $\mathcal{O}(\Pi)$ such that for any two functionally equivalent programs Π_0, Π_1 (that is, their outputs and run-time are the same on all inputs,) $\mathcal{O}(\Pi_0)$ is indistinguishable from $\mathcal{O}(\Pi_1)$. *iO* for Turing machines [12, 32, 1] additionally requires that the size of the obfuscated code does not grow (more than polylogarithmically) with the running-time of the Turing machine.

We may also consider a useful strengthening of this notion—which we call “puncturable *iO*”—which, roughly speaking, requires indistinguishability of $\mathcal{O}(\Pi_0)$ and $\mathcal{O}(\Pi_1)$ as long as Π_0 and Π_1 differ *on at most one input* x^* and their outputs on input x^* are indistinguishable. More precisely, we say that a distribution D is *admissible* if there exists some x^* such that a) for every triple (Π_0, Π_1, Π) in the support of D , and every $x \neq x^*$, it holds that $\Pi_0(x) = \Pi_1(x) = \Pi(x)$, and b) $(\Pi, \Pi_0(x^*))$ and $(\Pi, \Pi_1(x^*))$ are computationally indistinguishable when (Π_0, Π_1, Π) are sampled randomly from D . Puncturable *iO* requires indistinguishability of $\mathcal{O}(\Pi_0)$ and $\mathcal{O}(\Pi_1)$ for Π_0, Π_1 sampled from any admissible distribution. Interestingly, for the case of *circuits*, puncturable *iO* is equivalent to (standard) *iO*.³ Indeed, such a notion is implicitly used in the beautiful and powerful punctured-program paradigm by Sahai and Waters [116], and all its applications. (In this context, think of Π as the “punctured” version of the programs Π_0, Π_1 .)

In the case of Turing machines, when restricting to the degenerate case of

³To see this, consider a hybrid program $\Pi^y(x)$ that runs $\Pi(x)$ if $x \neq x^*$ and otherwise (i.e., if $x = x^*$ outputs y). By the *iO* property we have that for every Π, Π_0, Π_1 in the support of D , $\mathcal{O}(\Pi^{\Pi_b(x^*)})$ is indistinguishable from $\mathcal{O}(\Pi_b)$. Thus, if $\mathcal{O}(\Pi_0), \mathcal{O}(\Pi_1)$ are distinguishable, so are $\mathcal{O}(\Pi^{\Pi_0(x^*)}), \mathcal{O}(\Pi^{\Pi_1(x^*)})$, which contradicts indistinguishability of $(\Pi, \Pi_0(x^*))$ and $(\Pi, \Pi_1(x^*))$.

Turing machines with no inputs (or more precisely, we only consider the execution of $\Pi()$ on the "empty" input), the notion of iO for Turing machines is equivalent to the notion of a compact *weak* RE. Compact RE, on the other hand, is equivalent to puncturable iO for Turing machines (without inputs). (Jumping ahead, as we shall see, for the case of Turing machines it is unlikely that puncturable iO is equivalent to standard iO .)

4.1.1 Our results

iO from sublinear RE We start by showing that sublinear RE is an extremely useful primitive: Subexponentially-secure sublinear RE implies indistinguishability obfuscators for all polynomial-size circuits.

Theorem 1. *The existence of subexponentially-secure sublinear RE and one-way functions implies the existence of subexponentially-secure iO for circuits.*

Before continuing, let us mention that Theorem 1 is related to a recent beautiful result by Ananth and Jain [3] which shows that *under the LWE assumption*, subexponentially-secure compact RE (satisfying only the weak indistinguishability security) implies iO for circuits. Their construction goes from RE to *functional encryption* (FE) [26], and then from FE to iO ; (the first step relies on previous constructions of FE [74, 78], while the second step relies on a sequence of complex transformations and analysis). In contrast, the proof of Theorem 1 directly constructs iO from RE in a surprisingly simple way: We essentially use the GGM construction [67] that builds a PRF from a PRG using a tree, but replace the PRG with a RE. Let us explain in more details below.

Consider a program Π taking n -bit inputs. We consider a binary tree where the leaves are randomized encodings of the function applied to all possible inputs, and each node in the tree is a randomized encoding that generates its two children. More precisely, given a sequence of bits x_1, \dots, x_i , let $\tilde{\Pi}_{R,x_1,\dots,x_i}$ denote an (input-less) program that

- if $i = n$ simply outputs a RE of the program Π and input (x_1, \dots, x_n) using R as randomness, and
- otherwise, after expanding R_0, R_1, R_2, R_3 from R using a PRG, outputs randomized encodings of (input-less) programs $\tilde{\Pi}_{R_0,x_1,\dots,x_i,0}$ and $\tilde{\Pi}_{R_1,x_1,\dots,x_i,1}$ using respectively R_2, R_3 as randomness.

We associate each node in the binary tree that has index x_1, \dots, x_i with a randomized encoding of the program $\tilde{\Pi}_{R,x_1,\dots,x_i}$, denoted as $\hat{\Pi}_{R,x_1,\dots,x_i}$. In particular, the root of the tree is associated with a randomized encoding $\hat{\Pi}$ of the (initial) program $\tilde{\Pi}_R$ hardwired with a randomly chosen R .

The obfuscation of Π is now a program with the “root” $\hat{\Pi}$ hardcoded, and given an input x , computes the path from the root to the leaf x – by recursively evaluating the randomized encodings associated with nodes on the path – and finally outputs the evaluation of the leaf. More precisely, on input x , evaluate $\hat{\Pi}$ to obtain $\hat{\Pi}_0, \hat{\Pi}_1$, next evaluate $\hat{\Pi}_{x_1}$ to obtain $\hat{\Pi}_{x_1,0}, \hat{\Pi}_{x_1,1}$, so on and so forth until $\hat{\Pi}_{x_1,\dots,x_n}$ is evaluated, yielding the output $\Pi(x_1, \dots, x_n)$.

Note that for any two functionally equivalent programs, the randomized encodings associated with individual leaf node are computationally indistinguishable by the indistinguishability security property (the non-distributional version suffices here). Then, by the distributional indistinguishability security,

the randomized encodings associated with tree nodes one layer above are also indistinguishable. Thus, by induction, it follows that the roots are indistinguishable, which implies that obfuscations of functionally equivalent programs are indistinguishable. Let us note that the reason that subexponential security is needed is that each time we go up one level in the tree (in the inductive argument), we lose at least a factor 2 in the indistinguishability gap (as each node generates two randomized encodings, its children). Hence, we need to ensure that encodings are at least $\text{poly}(2^n)$ -indistinguishable, which can be done by scaling up the security parameter.

On the existence of Compact and Sublinear RE We next turn to investigating the existence of compact and sublinear RE. We show—assuming just the existence of subexponentially-secure one-way functions—*impossibility* of subexponentially-secure sublinear (and thus also compact) RE.⁴

Theorem 2. *Assume the existence of subexponentially secure one-way functions. Then, there do not exist subexponentially-secure sublinear RE.*

As observed above, compact RE can be interpreted as a stronger notion of iO (which we referred to as *puncturable iO*) for “degenerate” input-less Turing machines, and as such Theorem 2 rules out (assuming just one-way functions) such a natural strengthening of iO for (input-less) Turing machines. We note that this impossibility stands in contrast with the case of *circuits* where puncturable iO is equivalent to iO .

We remark that although it may seem like Theorem 2 makes Theorem 1 pointless, it turns out that Theorem 1 plays a crucial role in the proof of Theorem

⁴This result was established after hearing that Bitansky and Paneth had ruled out compact RE assuming public-coin differing-input obfuscation for Turing Machines and collision-resistant hashfunctions. We are very grateful to them for informing us of their result.

2: Theorem 2 is proven by first ruling out sublinear (even just polynomially-secure) RE *assuming* iO and one-way functions. Next, by using Theorem 1, the iO assumption comes for free if considering subexponentially-secure RE. That is, assuming one-way functions, we have the following paradigm:

$$\begin{aligned} \text{exp secure sublinear RE} &\stackrel{\text{Theorem 1}}{\implies} iO \\ &\implies \text{impossibility of (poly secure) sublinear RE} \end{aligned}$$

Let us now briefly sketch how to rule out sublinear RE assuming iO and one-way functions (as mentioned, Theorem 2 is then deduced by relying on Theorem 1). The idea is somewhat similar to the non-black-box zero-knowledge protocol of Barak [7].

Let $\Pi_{s,u}^b$ be a program that takes no input and outputs a sufficiently long pseudo-random string $y = \text{PRG}(s)$ and an indistinguishability obfuscation \tilde{R}_y^b (generated using pseudo-random coins $\text{PRG}(u)$) of the program R_y^b . The program R_y^b takes input Σ of length $|y|/2$, and outputs b iff Σ , when interpreted as an input-less Turing machine, generates y ; in all other cases, it outputs \perp .⁵ We note that the size of the program $\Pi_{s,u}^b$ is linear in the security parameter λ , whereas the pseudo-random string y it generates could have length $|y| = \lambda^\alpha$ for any sufficiently large constant α .

Consider the pair of distributions $\Pi_{U_\lambda, U_\lambda}^0$ and $\Pi_{U_\lambda, U_\lambda}^1$ that samples respectively programs $\Pi_{s,u}^0$ and $\Pi_{s,u}^1$ as described above with random s and u . We first argue that their outputs are computationally indistinguishable. Recall that the output of $\Pi_{s,u}^b$ is a pair (y, \tilde{R}_y^b) . By the pseudorandomness of PRG, this output distribution is indistinguishable from (X, \tilde{R}_X^b) where X a uniformly distributed random

⁵To enable this, we require iO for bounded-input Turing machines, whereas Theorem 1 only gives us iO for circuits. However, by the results of [19, 43, 90] we can go from iO for circuits to iO for bounded-inputs Turing machines.

variable over λ^α bit strings. With overwhelming probability X has high Kolmogorov complexity, and when this happens R_X^b is functionally equivalent to the program R_\perp that always outputs \perp . Therefore, by the security of the iO , the output of programs sampled from $\Pi_{U_\lambda, U_\lambda}^b$ is computationally indistinguishable to (X, \tilde{R}_\perp) , and hence outputs of $\Pi_{U_\lambda, U_\lambda}^0$ and $\Pi_{U_\lambda, U_\lambda}^1$ are indistinguishable.

Let us now turn to showing that randomized encodings of $\Pi_{U_\lambda, U_\lambda}^0$ and $\Pi_{U_\lambda, U_\lambda}^1$ can be distinguished. Recall that a randomized encoding $\hat{\Pi}^b$ of $\Pi_{U_\lambda, U_\lambda}^b$ itself can be viewed as a (input-less) program that outputs (y, \tilde{R}_y^b) . Given $\hat{\Pi}^b$, the distinguisher can thus first evaluate $\hat{\Pi}^b$ to obtain (y, \tilde{R}_y^b) and next evaluate $\tilde{R}_y^b(\hat{\Pi}^b)$ to attempt to recover b . Note that $\hat{\Pi}^b$ clearly is a program that generates y (as its first input); furthermore, if the RE scheme is compact, the length of the program $|\hat{\Pi}^b|$ is bounded by $\text{poly}(\lambda, \log \lambda^\alpha)$, which is far smaller than $|y|/2 = \lambda^\alpha/2$ when α is sufficiently large. Therefore, $\Sigma = \hat{\Pi}^b$ is indeed an input that makes \tilde{R}_y^b output b , enabling the distinguisher to distinguish $\hat{\Pi}^0$ and $\hat{\Pi}^1$ with probability close to 1!

Finally, if the RE is only sublinear, the length of the encoding $|\hat{\Pi}^b|$ is only sublinear in the output length, in particular, bounded by $\text{poly}(\lambda)(\lambda^\alpha)^{1-\epsilon}$ for some constant $\epsilon > 0$. If $\alpha > 1/(1 - \epsilon)$ (which clearly happens if ϵ is sufficiently small), then we do not get enough “compression” for the above proof to go through. We circumvent this problem by composing a sublinear RE with itself a sufficient (constant) number of times—to compose once, consider creating randomized encoding of the randomized encoding of a function, instead of the function itself; each time of composition reduces the size of the encoding to be w.r.t. a smaller exponent $1 - \epsilon'$. Therefore, it is without loss of generality to assume that ϵ is any sufficiently *big* constant satisfying $\alpha \ll 1/(1 - \epsilon)$; so the desired compression occurs.

Sublinear RE in the CRS model from sublinear FE Despite Theorem 2, not all is lost. We remark that any sublinear functional encryption scheme (FE) [3, 22] almost directly yields a sublinear RE in the *Common Reference String (CRS)* model; roughly speaking, an FE scheme is called sublinear if the encryption time is sublinear in the size of the circuit that can be evaluated on the encrypted message.

Theorem 3. *Assume the existence of subexponentially-secure sublinear (resp. compact) FE. Then there exists a subexponentially-secure sublinear (resp. compact) RE in the CRS model.*

Furthermore, Theorem 1 straightforwardly extends also to RE in the CRS model. Taken together, these results provide an alternative, modular, simpler proof of the recent results of Ananth and Jain [3] and Bitansky and Vaikuntanathan [22] showing that subexponentially-secure sublinear FE implies subexponentially-secure iO . (All these approaches, including a related work by Brakerski, Komargodski and Segev [35] have one thing in common though: they all proceed by processing inputs one bit at a time, and hard-coding parts of input to the program.)

Theorem 4 (informal, alternative proof of [22, 3]). *Assume the existence of subexponentially-secure sublinear FE. Then there exists a subexponentially-secure iO for circuits.*

But there are also other ways to instantiate sublinear RE in the CRS model. We show that under the *subexponential LWE assumption* (relying on [73, 2, 78]) sublinear RE in the CRS model can be based on a significantly weaker notion of sublinear FE—namely FE schemes where the encryption time may be fully polynomial (in the size of the circuit to be evaluated) but only the *size of the ciphertext*

is sublinear in the circuit size—we refer to this notion as a *FE with sublinear ciphertexts*. Roughly speaking, we show this by 1) transforming the “succinct” FE (i.e. compact FE for 1-bit outputs) of [73, 2] into an RE which depends linearly on the output length but only polylogarithmically on the running time, 2) transforming an FE with sublinear ciphertext into an RE with “large” running-time but short output, and 3) finally composing the two randomized encodings (i.e. computing the step 1 RE of the step 2 RE).

Combining this result with (the CRS-extended version of) Theorem 1, we get:

Theorem 5 (informal). *Assume the existence of subexponentially-secure FE with sublinear ciphertexts and the subexponential LWE assumption. Then there exists a subexponentially-secure iO for circuits.*

Toward Turing Machine Obfuscation with Unbounded Inputs We finally address the question of constructing indistinguishability obfuscators for Turing machines with *unbounded* inputs. (For the case of Turing machine obfuscation with unbounded-length inputs, the same obfuscated code needs to work for every input-length, and in particular, the size of the obfuscated code cannot grow with it.) Although it is known that subexponentially secure iO for circuits implies iO for Turing machines with *bounded inputs lengths* [19, 43, 90], the only known construction of iO for Turing machines with unbounded inputs relies on (public-coin) differing-input obfuscation for circuits and (public-coin) SNARKs [32, 1, 85]—these are strong “extractability” assumptions (and variants of them are known to be implausible [18, 58, 21]).

We note that the construction from Theorem 1 easily extends to show that

subexponentially-secure *compact* RE implies *iO* for Turing machines with unbounded input: instead of having a binary tree, we have a ternary tree where the “third” child of a node is always a leaf; that is, for a tree node corresponding to x_1, \dots, x_i , its third child is associated with a randomized encoding of program Π , and input (x_1, \dots, x_i) , which can be evaluated to obtain output $\Pi(x_1, \dots, x_i)$. Then, by using a tree of *super-polynomial* depth, we can handle any polynomial-length input. Note that since obfuscating a program only involves computing the root RE (as before), the obfuscation is still efficient. Moreover, for any input, we still compute the output of the program in time polynomial in the length of the input by evaluating the “third” child of the node when all input bits have been processed.⁶

But as shown in Theorem 2, compact RE cannot exist (assuming one-way functions)! However, just as for the case of differing-inputs obfuscation and SNARKs, we may assume the existence of compact RE for *restricted* types of “nice” distributions (over programs and inputs), for which impossibility does not hold, yet the construction in Theorem 1 still works. We formalize one natural class of such distributions, and may assume that the *iO* for bounded-input Turing machines construction of [90] (based on *iO* for circuits) yields such a compact RE (for the restricted class of distributions). This yields a new candidate construction of unbounded input Turing machines (based on a very different type of assumption than known constructions).

⁶Proving security becomes slightly more problematic since there is no longer a polynomial bound on the depth of the tree (recall that we required $\text{poly}(2^n)$ -indistinguishable RE to deal with inputs of length n). This issue, however, can be dealt with by using larger and larger security parameters for RE that are deeper down in the tree.

4.2 Preliminaries

Let \mathcal{N} denote the set of positive integers, and $[n]$ denote the set $\{1, 2, \dots, n\}$. We denote by PPT probabilistic polynomial time Turing machines. The term *negligible* is used for denoting functions that are (asymptotically) smaller than one over any polynomial. More precisely, a function $\nu(\cdot)$ from non-negative integers to reals is called *negligible* if for every constant $c > 0$ and all sufficiently large n , it holds that $\nu(n) < n^{-c}$.

Turing machine notation For any Turing machine Π , input $x \in \{0, 1\}^*$ and time bound $T \in \mathbb{N}$, we denote by $\Pi^T(x)$ the output of Π on x when run for T steps. We refer to $\{\mathcal{M}_\lambda\}_{\lambda \in \mathbb{N}}$ as a class of Turing machines. One particular class we will consider is the class of Turing machines that have 1-bit output. We call such a machine a Boolean Turing machine. Throughout this paper, by *Turing machine* we refer to a machine with *multi-bit* output unless we explicitly mention it to be a Boolean Turing machine.

4.2.1 Concrete Security

Definition 1 ($(\lambda_0, S(\cdot))$ -indistinguishability). *A pair of distributions X, Y are S -indistinguishable for some $S \in \mathbb{N}$ if every S -size distinguisher D it holds that*

$$|\Pr[x \xleftarrow{\$} X : D(x) = 1] - \Pr[y \xleftarrow{\$} Y : D(y) = 1]| \leq \frac{1}{S}$$

A pair of ensembles $\{X_\lambda\}, \{Y_\lambda\}$ are $(\lambda_0, S(\cdot))$ -indistinguishable for some $\lambda_0 \in \mathbb{N}$ and $S : \mathbb{N} \rightarrow \mathbb{N}$ if for every security parameter $\lambda > \lambda_0$, the distributions X_λ and Y_λ are $S(\lambda)$ indistinguishable.

Discussion on $(\lambda_0, S(\cdot))$ -indistinguishability: We remark that the above definition requires that there is a universal λ_0 that works for all distinguisher D . A seemingly weaker variant could switch the order of quantifiers and only require that for every distinguisher D there is a λ_0 . We show that the above definition is w.l.o.g, since it is implied by the following standard definition with auxiliary inputs in the weaker fashion.

Let U be a universal TM that on an input x and a circuit C computes $C(x)$. Let $S' : N \rightarrow N$ denote the run time $S'(S)$ of U on input a size S circuit.

Definition 2. *A pair of ensembles $\{X_\lambda\}, \{Y_\lambda\}$ are $S(\cdot)$ -indistinguishable if for every $S' \circ S(\cdot)$ -time uniform TM distinguisher D , there exists a $\lambda_0 \in N$, such that, for every security parameter $\lambda > \lambda_0$, and every auxiliary input $z = z_\lambda \in \{0, 1\}^*$,*

$$|\Pr[x \stackrel{\$}{\leftarrow} X_\lambda : D(1^\lambda, x, z) = 1] - \Pr[y \stackrel{\$}{\leftarrow} Y_\lambda : D(1^\lambda, y, z) = 1]| \leq \frac{1}{S(\lambda)}$$

This definition implies $(\lambda_0, S(\cdot))$ -indistinguishability. Consider a distinguisher D that on input $(1^\lambda, x, z)$ runs the universal TM $U(x, z)$, and let λ_U be the constant associated with it. For any $\lambda > \lambda_U$, and every $S(\lambda)$ -size circuit C , by setting the auxiliary input $z = C$, the above definition implies that the distinguishing gap by C is at most $1/S(\lambda)$. Therefore, λ_U is effectively the universal constant that works for all (circuit) distinguisher.

Below, we state definitions of cryptographic primitives using $(\lambda_0, S(\cdot))$ indistinguishability. Traditional polynomial or sub-exponential security can be directly derived from such more concrete definitions as follows:

Definition 3 (Polynomial Indistinguishability). *A pair of ensembles $\{X_\lambda\}, \{Y_\lambda\}$ are polynomially indistinguishable if for every polynomial $p(\cdot)$, there is a constant $\lambda_p \in N$, such that, the two ensembles are $(\lambda_p, p(\cdot))$ -indistinguishable.*

Definition 4 (Sub-exponential Indistinguishability). *A pair of ensembles $\{X_\lambda\}, \{Y_\lambda\}$ are sub-exponentially indistinguishable, if there is a sub-exponential function $S(\lambda) = 2^{\lambda^\varepsilon}$ with $\varepsilon \in (0, 1)$ and a constant $\lambda_0 \in \mathbb{N}$, such that, the two ensembles are $(\lambda_0, S(\cdot))$ -indistinguishable.*

4.2.2 Standard cryptographic primitives

Definition 5 (Pseudorandom Generator). *A deterministic PT uniform machine PRG is a pseudorandom generator if the following conditions are satisfied:*

Syntax *For every $\lambda, \lambda' \in \mathbb{N}$ and every $r \in \{0, 1\}^\lambda$, $\text{PRG}(r, \lambda')$ outputs $r' \in \{0, 1\}^{\lambda'}$*

$(\lambda_0, S(\cdot))$ -Security *For every function $p(\cdot)$, such that, $p(\lambda) \leq S(\lambda)$ for all λ , the following ensembles are $(\lambda_0, S(\cdot))$ indistinguishable*

$$\left\{ r \stackrel{s}{\leftarrow} \{0, 1\}^\lambda : \text{PRG}(r, p(\lambda)) \right\} \left\{ r' \stackrel{s}{\leftarrow} \{0, 1\}^{p(\lambda)} \right\}$$

4.2.3 Indistinguishability Obfuscation

In this section, we recall the definition of indistinguishability obfuscation for Turing machines from [12, 32, 1]. Following [32], we consider two notions of obfuscation for Turing machines. The first definition, called *bounded-input* indistinguishability obfuscation, only requires the obfuscated program to work for inputs of *bounded length* and furthermore the size of the obfuscated program may depend polynomially on this input length bound. (This is the notion achieved in [19, 43, 90] assuming subexponentially-secure *iO* for circuits and one-way functions.)

The second notion considered in [32] is stronger and requires the obfuscated program to work on any arbitrary polynomial length input (and the size of the obfuscated machine thus only depends on the program size and security parameter). We refer to this notion as *unbounded-input* indistinguishability obfuscation. (This stronger notion of unbounded-input indistinguishability obfuscator for Turing machines is only known to be achievable based on strong “extractability assumptions”—namely, (public-coin) differing-input obfuscation for circuits and (public-coin) SNARKs [32, 1, 85], variants of which are known to be implausible [18, 58, 21]).

Definition 6 (Indistinguishability Obfuscator (*iO*) for a class of Turing machines). *An indistinguishability obfuscator for a class of Turing machines $\{\mathcal{M}_\lambda\}_{\lambda \in \mathbb{N}}$ is a uniform machine that behaves as follows:*

$\hat{\Pi} \leftarrow iO(1^\lambda, \Pi, T)$: *iO takes as input a security parameter 1^λ , the Turing machine to obfuscate $\Pi \in \mathcal{M}_\lambda$ and a time bound T for Π . It outputs a Turing machine $\hat{\Pi}$.*

We require the following conditions to hold.

Correctness: *For every $\lambda \in \mathbb{N}$, $\Pi_\lambda \in \mathcal{M}_\lambda$, input x_λ and time bound T_λ ,*

$$\Pr[(\tilde{\Pi} \stackrel{s}{\leftarrow} iO(1^\lambda, \Pi_\lambda, T_\lambda) : \tilde{\Pi}(x_\lambda) = \Pi^T(x_\lambda))] = 1 .$$

Efficiency: *The running times of iO and $\hat{\Pi}$ are bounded as follows:*

There exists polynomial p such that for every security parameter λ , Turing machine $\Pi \in \mathcal{M}_\lambda$, time bound T and every obfuscated machine $\hat{\Pi} \leftarrow iO(1^\lambda, \Pi, T)$ and input x , we have that

$$\text{Time}_{iO}(1^\lambda, \Pi, T) \leq p(\lambda, |\Pi|, \log T)$$

$$\text{Time}_{\hat{\Pi}}(x) \leq p(\lambda, |\Pi|, |x|, T)$$

$(\lambda_0, S(\cdot))$ -**Security**: For every ensemble of pairs of Turing machines and time bounds $\{\Pi_{0,\lambda}, \Pi_{1,\lambda}, T_\lambda\}$ where for every $\lambda \in \mathbb{N}$, $\Pi_0 = \Pi_{0,\lambda}$, $\Pi_1 = \Pi_{1,\lambda}$, $T = T_\lambda$, satisfying the following

$$\begin{aligned} \Pi_0, \Pi_1 \in \mathcal{M}_\lambda \quad |\Pi_0| = |\Pi_1| \leq \text{poly}(\lambda) \quad T \leq \text{poly}(\lambda) \\ \forall x, \Pi_0^T(x) = \Pi_1^T(x), \end{aligned}$$

the following ensembles are $(\lambda_0, S(\cdot))$ -indistinguishable

$$\{i\mathcal{O}(1^\lambda, \Pi_{0,\lambda}, T_\lambda)\} \{i\mathcal{O}(1^\lambda, \Pi_{1,\lambda}, T_\lambda)\} .$$

Definition 7 (Unbounded-input indistinguishability obfuscator for Turing machines). An unbounded-input indistinguishability obfuscator for Turing machines $i\mathcal{O}(\cdot, \cdot, \cdot)$ is simply an indistinguishability obfuscator for the class of all Boolean Turing machines.

Remark 1 (Obfuscation for Boolean Turing machines is without loss of generality). The above definition is equivalent to one that considers the class of all Turing machines. Any Turing machine with output length m can be represented as a Boolean Turing machine that takes in an additional input $i \in [m]$ and returns the i^{th} bit of the m -bit long output.

Definition 8 (Bounded-input indistinguishability obfuscator for Turing machines). A bounded-input indistinguishability obfuscator for Turing machines $i\mathcal{O}(\cdot, \cdot, \cdot, \cdot)$ is a uniform machine such that for every polynomial p , $i\mathcal{O}(p, \cdot, \cdot, \cdot)$ is an indistinguishability obfuscator for the class of Turing machines $\{\mathcal{M}_\lambda\}$ where \mathcal{M}_λ are machines that accept only inputs of length $p(\lambda)$. Additionally, $i\mathcal{O}(p, 1^\lambda, \Pi, T)$ is allowed to run in time $\text{poly}(p(\lambda) + \lambda + |\Pi| + \log T)$.

Finally, we define weaker variants of the above definitions where the size of the obfuscated program is sub-linear (instead of poly-logarithmic) in the time bound.

Definition 9 (Sub-linear efficiency for Indistinguishability Obfuscators). *We say an indistinguishability obfuscator iO for a class of Turing machines $\{\mathcal{M}_\lambda\}$ has sub-linear efficiency if it satisfies the requirements in Definition 6 with the efficiency requirement replaced with the following.*

Efficiency: *The running times of iO and $\hat{\Pi}$ are bounded as follows:*

There exists polynomial p and constant $\epsilon > 0$ such that for every security parameter λ , Turing machine $\Pi \in \mathcal{M}_\lambda$, time bound T and every obfuscated machine $\hat{\Pi} \leftarrow iO(1^\lambda, \Pi, T)$ and input x , we have that

$$\text{Time}_{iO}(1^\lambda, \Pi, T) \leq p(\lambda, |\Pi|)T^{1-\epsilon}$$

$$\text{Time}_{\hat{\Pi}}(x) \leq p(\lambda, |\Pi|, |x|, T)$$

4.2.4 Functional Encryption

Definition 10 (Selectively-secure Single-Query Public-key Functional Encryption). *A tuple of PPT algorithms (FE.Setup, FE.Enc, FE.Dec) is a selectively-secure functional encryption scheme for a class of circuits $\{C_\lambda\}$ if it satisfies the following properties.*

Completeness *For every $\lambda \in \mathbb{N}$, $C \in C_\lambda$ and message $m \in \{0, 1\}^*$,*

$$\Pr \left[\begin{array}{l} (mpk, msk) \leftarrow \text{FE.Setup}(1^\lambda) \\ c \leftarrow \text{FE.Enc}(1^\lambda, m) \quad : C(m) \leftarrow \text{FE.Dec}(sk_C, c) \\ sk_C \leftarrow \text{FE.KeyGen}(msk, C) \end{array} \right] = 1$$

$(\lambda_0, S(\cdot))$ -Selective-security *For every ensemble of circuits and pair of messages $\{C_\lambda, m_{0,\lambda}, m_{1,\lambda}\}$ where $C_\lambda \in C_\lambda$, $|C_\lambda|, |m_{0,\lambda}|, |m_{1,\lambda}| \leq \text{poly}(\lambda)$, and $C_\lambda(m_{0,\lambda}) =$*

$C_\lambda(m_{1,\lambda})$, the following ensembles of distributions $\{D_{0,\lambda}\}$ and $\{D_{1,\lambda}\}$ are $(\lambda_0, S(\cdot))$ -indistinguishable.

$$D_{b,\lambda} = \left(\begin{array}{l} (mpk, msk) \leftarrow \text{FE.Setup}(1^\lambda) \\ c \leftarrow \text{FE.Enc}(1^\lambda, m_{b,\lambda}) \\ sk_C \leftarrow \text{FE.KeyGen}(msk, C_\lambda) \end{array} : mpk, c, sk_C \right)$$

We note that in this work, we only need the security of the functional encryption scheme to hold with respect to statically chosen challenge messages and functions.

Definition 11 (Compact Functional Encryption). *We say a functional encryption scheme is compact if it additionally satisfies the following requirement:*

Compactness *The running time of FE.Enc is bounded as follows.*

There exists a polynomial p such that for every security parameter $\lambda \in \mathbb{N}$ and message $m \in \{0, 1\}^$, $\text{Time}_{\text{FE.Enc}}(1^\lambda, m) \leq p(\lambda, |m|, \text{polylog}(s))$, where $s = \max_{C \in \mathcal{C}_\lambda} |C|$.*

Furthermore, we say the functional encryption scheme has sub-linear compactness if there exists a polynomial p and constant $\epsilon > 0$ such that for every security parameter $\lambda \in \mathbb{N}$ and message $m \in \{0, 1\}^$, $\text{Time}_{\text{FE.Enc}}(1^\lambda, m) \leq p(\lambda, |m|)s^{1-\epsilon}$.*

We also define a notion of succinctness, as follows:

Definition 12 (Succinct Functional Encryption). *A compact functional encryption scheme for a class of circuits that output only a single bit is called a succinct functional encryption scheme.*

Theorem 6 ([73]). *Assuming (sub-exponentially secure) LWE, there exists a (sub-exponentially secure) succinct functional encryption scheme for NC^1 .*

We note that [73] do not explicitly consider sub-exponentially secure succinct functional encryption, but their construction satisfies it (assuming sub-exponentially secure LWE).

Theorem 7 ([73, 2]). *Assuming the existence of symmetric-key encryption with decryption in NC^1 (resp. sub-exponentially secure) and succinct FE for NC^1 (resp. sub-exponentially secure), there exists succinct FE for P/poly (resp. sub-exponentially secure).*

We also consider an even weaker notion of sublinear-compactness, where only the ciphertext size is sublinear in the size bound s of the function being evaluation, but the encryption time can depend polynomially on s .

Definition 13 (Weakly Sublinear Compact Functional Encryption). *We say a functional encryption scheme for a class of circuits $\{C_\lambda\}$ is weakly sublinear compact if there exists $\epsilon > 0$ such that for every $\lambda \in \mathbb{N}$, $pk \leftarrow \text{FE.Setup}(1^\lambda)$ and $m \in \{0, 1\}^*$ we have that*

$$\begin{aligned} \text{Time}_{\text{FE.Enc}}(pk, m) &= \text{poly}(\lambda, |m|, s) \\ \text{outlen}_{\text{FE.Enc}}(pk, m) &= s^{1-\epsilon} \cdot \text{poly}(\lambda, |m|) \end{aligned}$$

where $s = \max_{C \in C_\lambda} |C|$.

4.3 Randomized Encoding Schemes

Roughly speaking, randomized encoding schemes encodes a computation of a program Π on an input x , into an encoded computation $(\hat{\Pi}, \hat{x})$, with the following two properties: First, the encoded computation evaluates to the same output

$\Pi(x)$, while leaking no other information about Π and x . Second, the encoding is “simpler” to compute than the original computation. In the literature, different measures of simplicity have been considered. For instance, in the original works by [83, 6], the depth of computation is used and it was shown that any computation in P can be encoded in NC_1 using Yao’s garbled circuits [118]. A recent line of works [19, 43, 90] uses the time-complexity as the measure and show that any *Boolean* Turing machine computation can be encoded in time poly-logarithmic in the run-time of the computation.

Traditionally, the security of randomized encoding schemes are captured via simulation. In this work, we consider a new *distributional* indistinguishability-based security notion, and show that it is implied by the transitional simulation security. Additionally, we further explore how compact the encoded computation can be: Similar to the recent works [19, 43, 90], we consider encoding whose size depends poly-logarithmically on the run-time of the encoded computation; but differently, we directly consider Turing machines with arbitrary length outputs, and require the size of the encoding to be independent of the output length. Such scheme is called a compact randomized encoding scheme.

4.3.1 Randomized Encoding with Simulation Security

In this section, we recall the traditional definition of randomized encoding with simulation security [83, 6].

Definition 14 (Randomized Encoding Scheme for a Class of Turing Machines). *A Randomized Encoding scheme RE for a class of Turing machines $\{\mathcal{M}_\lambda\}$ consists of two algorithms,*

- $(\hat{\Pi}, \hat{x}) \stackrel{s}{\leftarrow} \text{Enc}(1^\lambda, \Pi, x, T)$: On input a security parameter 1^λ , Turing machine $\Pi \in \mathcal{M}_\lambda$, input x and time bound T , Enc generates an encoded machine $\hat{\Pi}$ and encoded input \hat{x} .
- $y = \text{Eval}(\hat{\Pi}, \hat{x})$: On input $(\hat{\Pi}, \hat{x})$ produced by Enc , Eval outputs y .

Correctness: The two algorithms Enc and Eval satisfy the following correctness condition: For all security parameters $\lambda \in \mathbb{N}$, Turing machines $\Pi \in \mathcal{M}_\lambda$, inputs x and time bounds T , it holds that,

$$\Pr[(\hat{\Pi}, \hat{x}) \stackrel{s}{\leftarrow} \text{Enc}(1^\lambda, \Pi, x, T) : \text{Eval}(\hat{\Pi}, \hat{x}) = \Pi^T(x)] = 1$$

Definition 15 ($(\lambda_0, S(\cdot))$ -Simulation Security). A randomized encoding scheme RE for a class of Turing machines $\{\mathcal{M}_\lambda\}$ satisfies $(\lambda_0, S(\cdot))$ -**simulation security**, if there exists a PPT algorithm Sim and a constant c , such that, for every polynomial B , and ensemble $\{\Pi_\lambda, x_\lambda, T_\lambda\}$ where $\Pi_\lambda \in \mathcal{M}_\lambda$ and $|\Pi_\lambda|, |x_\lambda|, T_\lambda \leq B(\lambda)$, the following ensembles are $(\lambda_0, S'(\lambda))$ indistinguishable with $S'(\lambda) = S(\lambda) - B(\lambda)^d$ for all $\lambda \in N$.

$$\left\{ (\hat{\Pi}, \hat{x}) \stackrel{s}{\leftarrow} \text{Enc}(1^\lambda, \Pi, x, T) : \hat{\Pi}, \hat{x} \right\} \\ \left\{ (\hat{\Pi}, \hat{x}) \stackrel{s}{\leftarrow} \text{Sim}(1^\lambda, \Pi^T(x), 1^{|\Pi|}, 1^{|x|}, T) : \hat{\Pi}, \hat{x} \right\}$$

where $\Pi = \Pi_\lambda$, $x = x_\lambda$, and $T = T_\lambda$.

A recent line of works [19, 43, 90] constructed randomized encoding with polynomial simulation security (i.e., the simulation is polynomially indistinguishable to the honest encoding in the above definition) for the class of *Boolean* Turing machines, where the time complexity of encoding is independent of the run-time of the Turing machine.

Theorem 8 (Simulation-Based Randomized Encoding for Boolean Turing Machines [90]). *Assuming the existence of indistinguishability obfuscation for circuits*

and injective pseudo-random generators, there is a randomized encoding scheme $\text{RE} = (\text{Enc}, \text{Eval})$ satisfying polynomial simulation security for the class of Boolean Turing machines, with the following efficiency:

- For every security parameter λ , Boolean Turing machine Π , input x , time bound T and every encoded pair $(\hat{\Pi}, \hat{x}) \leftarrow \text{Enc}(1^\lambda, \Pi, x, T)$, we have that

$$\text{Time}_{\text{Enc}}(1^\lambda, \Pi, x, T) = \text{poly}(\lambda, |\Pi|, |x|, \log T)$$

$$\text{Time}_{\text{Eval}}(\hat{\Pi}, \hat{x}) = \text{poly}(\lambda, |\Pi|, |x|, T)$$

In this paper, we consider the class of *all* Turing machines, including ones with arbitrarily long outputs. One can obtain a randomized encoding for Turing machines with ℓ -bit outputs, by encoding a collection of ℓ Turing machines each outputting one output bit, using a randomized encoding for Boolean Turing machines. It yields a scheme whose encoding time, as well as the size of encoding, scales linearly with the output length (and still poly-logarithmically with the run-time of the encoded computation). As we show later in the paper, this essentially is tight, meaning that there is certain computation (namely, evaluating pseudo-random generators) for which the size of the randomized encoding cannot be sub-linear in the output length. In other words, when simulation security is required, encoding has to be as long as the output length in general.

4.3.2 Distributional Indistinguishability Security

In this paper, we study randomized encoding for all Turing machine computation, whose encoding size is independent of the output length of the

computation—we say such randomized encoding schemes are **compact**. Towards this, we must consider weaker security notions than simulation security, and indistinguishability-based security notions are natural candidates. One weaker notion that has been considered in the literature requires encoding of two computation, (Π_1, x_1) and (Π_2, x_2) with the same output $\Pi_1(x_1) = \Pi_2(x_2)$, to be indistinguishable. In this work, we generalize this notion to, what called *distributional* indistinguishability security—this notion requires encoding of computations sampled from two distributions, $(\Pi_1, x_1) \stackrel{\$}{\leftarrow} D_1$ and $(\Pi_2, x_2) \stackrel{\$}{\leftarrow} D_2$, to be indistinguishable, provided that their outputs are indistinguishable.

Definition 16 (Distributional $(\lambda_0, S(\cdot))$ -Indistinguishability Security). *A randomized encoding scheme RE for a class of Turing machines $\{\mathcal{M}_\lambda\}$ satisfies distributional $(\lambda_0, S(\cdot))$ -indistinguishability security, (or $(\lambda_0, S(\cdot))$ -ind-security for short) if the following is true w.r.t. some constant $c > 0$:*

For every ensembles of distributions $\{D_{0,\lambda}\}$ and $\{D_{1,\lambda}\}$ with the following property:

1. *there exists a polynomial B , such that, for every $b \in \{0, 1\}$, $D_{b,\lambda}$ is a distribution over tuples of the form (Π_b, x_b, T_b) , where Π_b is a Turing machine, x_b is an input and T_b is a time bound, and $\lambda, |\Pi_b|, |x_b|, T_b \leq B(\lambda)$.*
2. *there exist an integer $\lambda'_0 \geq \lambda_0$, and a function S' with $S'(\lambda) \leq S(\lambda)$ for all λ , such that, the following ensembles of output distributions are $(\lambda'_0, S'(\cdot))$ -indistinguishable,*

$$\left\{ (\Pi_0, x_0, T_0) \stackrel{\$}{\leftarrow} \mathcal{D}_{0,\lambda} : \Pi_0^{T_0}(x_0), T_0, |\Pi_0|, |x_0| \right\}$$

$$\left\{ (\Pi_1, x_1, T_1) \stackrel{\$}{\leftarrow} \mathcal{D}_{1,\lambda} : \Pi_1^{T_1}(x_1), T_1, |\Pi_1|, |x_1| \right\}$$

the following ensembles of encoding is $(\lambda'_0, S''(\cdot))$ -indistinguishable, where $S''(\lambda) =$

$$\frac{S'(\lambda)}{\lambda^\varepsilon} - B(\lambda)^c.$$

$$\left\{ (\Pi_0, x_0, T_0) \stackrel{s}{\leftarrow} \mathcal{D}_{0,\lambda} : \text{Enc}(1^\lambda, \Pi_0, x_0, T_0) \right\}$$

$$\left\{ (\Pi_1, x_1, T_1) \stackrel{s}{\leftarrow} \mathcal{D}_{1,\lambda} : \text{Enc}(1^\lambda, \Pi_1, x_1, T_1) \right\}$$

For convenience, in the rest of the paper, we directly refer to distributional indistinguishability security as indistinguishability security. The above concrete security directly gives the standard polynomial and sub-exponential security.

Definition 17 (Polynomial and Sub-exponential Indistinguishability Security). *A randomized encoding scheme RE for a class of Turing machines $\{\mathcal{M}_\lambda\}$ satisfies polynomial ind-security, if it satisfies $(\lambda_p, p(\cdot))$ -indistinguishability security for every polynomial p and some $\lambda_p \in N$. Furthermore, it satisfies sub-exponential ind-security if it satisfies $(\lambda_0, S(\cdot))$ -indistinguishability security for $S(\lambda) = 2^{\lambda^\varepsilon}$ with some $\varepsilon \in (0, 1)$.*

We note that, by definition, it holds that any randomized encoding scheme that is $(\lambda_0, S(\cdot))$ -ind-secure, is also $(\lambda'_0, S'(\cdot))$ -ind-secure for any $\lambda'_0 \geq \lambda_0$ and S' s.t. $S'(\lambda) \leq S(\lambda)$ for every λ . Therefore, naturally, sub-exponential ind-security is stronger than polynomial ind-security.

Later, in Section 4.3.4, we show that RE schemes with ind-security are composable just as RE schemes with simulation security are. Additionally, in Section 4.3.5, we show that RE schemes satisfying simulation security also satisfy ind-security.

4.3.3 Compactness and Sublinear Compactness

With indistinguishability-security, we now define compact randomized encoding schemes for all Turing machines, whose time-complexity of encoding is in-

dependent of the output length.

Definition 18 (Compact Randomized Encoding for Turing machines). *A $(\lambda_0, S(\cdot))$ -ind-secure compact randomized encoding scheme for Turing machines, is a randomized encoding scheme with $(\lambda_0, S(\cdot))$ -indistinguishability security for the class of all Turing machines, with the following efficiency:*

- For every security parameter λ , Turing machine Π , input x , time bound T and every encoded pair $(\hat{\Pi}, \hat{x}) \leftarrow \text{Enc}(1^\lambda, \Pi, x, T)$, it holds

$$\text{Time}_{\text{Enc}}(1^\lambda, \Pi, x, T) = \text{poly}(\lambda, |\Pi|, |x|, \log T)$$

$$\text{Time}_{\text{Eval}}(\hat{\Pi}, \hat{x}) = \text{poly}(\lambda, |\Pi|, |x|, T)$$

In this work, we also consider a weaker variant of the above compactness requirement, where the encoding time is sub-linear (instead of poly-logarithmic) in the computation time. For our results a compact randomized encoding scheme with sub-linear efficiency will suffice.

Definition 19 (Sub-linear Compactness of Randomized Encoding schemes). *We say a randomized encoding scheme $\text{RE} = (\text{Enc}, \text{Eval})$ for a class of Turing machines $\{\mathcal{M}_\lambda\}$ has sub-linear compactness if the efficiency requirement on Enc in Definition 18 is relaxed to: For some constant $\varepsilon \in (0, 1)$,*

$$\text{Time}_{\text{Enc}}(1^\lambda, \Pi, x, T) \leq \text{poly}(\lambda, |\Pi|, |x|) \cdot T^{1-\varepsilon}$$

4.3.4 Composition of Ind-Security

It was shown in [84, 6] that randomized encoding schemes with simulation security are composable. We show that the same holds for indistinguishability

security. Let $RE_1 = (\text{Enc}_1, \text{Eval}_1)$ be a randomized encoding scheme for a class of Turing machines $\{\mathcal{M}_\lambda\}$, $RE_2 = (\text{Enc}_2, \text{Eval}_2)$ a randomized encoding for an appropriate class of Turing machines that includes the ensembles of functions $\{G[\lambda, \Pi, T, s]\}$ defined below, and PRG a pseudo-random generator. Consider the following composed randomized encoding scheme $(\text{Enc}, \text{Eval})$:

$$\begin{aligned} \text{Enc}(1^\lambda, \Pi, x, T) &: (\hat{G}, \hat{x}) \stackrel{s}{\leftarrow} \text{Enc}_2(1^\lambda, G, x, T_G(x)) \\ &\text{where } G[\lambda, \Pi, T, s](x) = \text{Enc}_1(1^\lambda, \Pi, x, T; \text{PRG}(s)) \\ \text{Eval}(\hat{G}, \hat{x}) &: (\hat{\Pi}, \hat{x}') = \text{Eval}_2(\hat{G}, \hat{x}), y = \text{Eval}_1(\hat{\Pi}, \hat{x}') \end{aligned}$$

where $T_G(x)$ is an upper bound on the run-time of G on input x , it can be efficiently calculated using an upper bound on the run-time of Enc_1 and PRG, in particular, $T_G(x) = \text{poly}(\text{Time}_{\text{Enc}_1}(1^\lambda, \Pi, x, T))$.

Lemma 1 (Composition). *If RE_1 is $(\lambda_1, S_1(\cdot))$ -ind-secure, and RE_2 and PRG are $(\lambda_2, S_2(\cdot))$ -ind-secure, with $\lambda_2 \leq \lambda_1$ and $S_2(\lambda) \geq S_1(\lambda)$. then $(\text{Enc}, \text{Eval})$ is $(\lambda_1, S_1(\cdot))$ -ind-secure.*

Proof. Let c_1 be the constant w.r.t. which the ind-security of RE_1 holds, and c_2 the constant for RE_2 . We show that the composed scheme $(\text{Enc}, \text{Eval})$ is $(\lambda_1, S_1(\cdot))$ -ind-secure w.r.t. some sufficiently large constant c , whose value would become clear in the proof below.

To show this, consider any pair of ensembles of distributions $\{D_{0,\lambda}\}$ and $\{D_{1,\lambda}\}$ that are (λ'_0, S') -indistinguishability for $\lambda'_0 \geq \lambda_1$ and $S'(\lambda) \leq S_1(\lambda)$ for all λ , and all parameters $\lambda, |\Pi_b|, |x_b|, T_b \leq B(\lambda)$ for some polynomial B . We need to show that encodings generated using Enc are (λ'_0, S'') -indistinguishable, where $S''(\lambda) \leq S'(\lambda)/\lambda^c - B(\lambda)^c$.

First, it follows directly from the indistinguishability security of RE_1 that the distributions H_1 and H_2 of encodings produced by Enc_1 are (λ'_0, S_1) -indistinguishable, where $S_H(\lambda) = S'(\lambda)/\lambda^{c_1} - B(\lambda)^{c_1}$,

$$H_1 = \left\{ (\Pi_0, x_0, T_0) \stackrel{\$}{\leftarrow} \mathcal{D}_{0,\lambda} : \text{Enc}_1(1^\lambda, \Pi_0, x_0, T_0) \right\}$$

$$H_2 = \left\{ (\Pi_1, x_1, T_1) \stackrel{\$}{\leftarrow} \mathcal{D}_{1,\lambda} : \text{Enc}_1(1^\lambda, \Pi_1, x_1, T_1) \right\}$$

Consider two modified distributions, where the encodings are generated using pseudo-random coins.

$$H_1^+ = \left\{ s \stackrel{\$}{\leftarrow} \{0, 1\}^\lambda, (\Pi_0, x_0, T_0) \stackrel{\$}{\leftarrow} \mathcal{D}_{0,\lambda} : \text{Enc}_1(1^\lambda, \Pi_0, x_0, T_0; \text{PRG}(s)) \right\}$$

$$H_2^+ = \left\{ s \stackrel{\$}{\leftarrow} \{0, 1\}^\lambda, (\Pi_1, x_1, T_1) \stackrel{\$}{\leftarrow} \mathcal{D}_{1,\lambda} : \text{Enc}_1(1^\lambda, \Pi_1, x_1, T_1; \text{PRG}(s)) \right\}$$

Since $\lambda, |\Pi_b|, |x_b|, T_b \leq B(\lambda)$, $\text{Time}_{\text{Enc}_1}(1^\lambda, \Pi_b, x_b, T_b)$ is bounded by $B(\lambda)^d$ for some constant d . Then, by the (λ_2, S_2) -indistinguishability of PRG, it follows that for every b no adversary of size $\tilde{S}_H(\lambda) = S_2(\lambda) - B(\lambda)^d$ can distinguish H_b^+ and H_b with probability larger than $1/S_2(\lambda)$. Therefore, (by a hybrid argument,) no adversary of size $\min(S_H(\lambda), \tilde{S}_H(\lambda))$ can distinguish H_1^+ and H_2^+ with probability larger than $2/\tilde{S}_H(\lambda) + 1/S_H(\lambda)$. Since $S'(\lambda) \leq S_1(\lambda) \leq S_2(\lambda)$, there is a $S_H^+(\lambda) = S'(\lambda)/\lambda^e - B(\lambda)^e$ with a sufficiently large constant e , such that, $\{H_1^+\}$ and $\{H_2^+\}$ are (λ'_0, S_H^+) -indistinguishable.

Consider the following ensembles of distributions $\{\mathcal{E}_{0,\lambda}\}$ and $\{\mathcal{E}_{1,\lambda}\}$:

$$\mathcal{E}_{b,\lambda} : s \stackrel{\$}{\leftarrow} \{0, 1\}^\lambda, (\Pi_b, x_b, T_b) \stackrel{\$}{\leftarrow} \mathcal{D}_{b,\lambda} \text{ output } (G_b, x_b, T'_b)$$

$$\text{where } G_b[\lambda, \Pi_b, T_b, s](x_b) = \text{Enc}_1(1^\lambda, \Pi_b, x_b, T_b; \text{PRG}(s)), T'_b = T_{G_b}(x_b).$$

where $T_{G_b}(x_b) = \text{poly}(\lambda, |\Pi_b|, |x_b|, \log T_b)$, determined by the run time of algorithms PRG and Enc_1 . By the indistinguishability of H_1^+ and H_2^+ , the output distributions of $\mathcal{E}_{0,\lambda}$ and $\mathcal{E}_{1,\lambda}$ are also (λ'_0, S_H^+) -indistinguishable. Moreover, all parameters $T'_b, |G_b|, |x_b|$ are bounded by $B(\lambda)^k$ for some constant k .

Therefore, it follows from the (λ_2, S_2) -security of RE_2 (and the fact that $\lambda_2 \leq \lambda_1 \leq \lambda'_0$, and $S_H^+(\lambda) \leq S_2(\lambda)$) that, encoding of G_0 and G_1 sampled from $\mathcal{E}_{0,\lambda}$ and $\mathcal{E}_{1,\lambda}$ are (λ'_0, S_3'') -indistinguishable, where $S''(\lambda) = S_H^+(\lambda)/\lambda^{c_2} - B(\lambda)^{kc_2} \geq S'(\lambda)/\lambda^c - B(\lambda)^c$ for a sufficiently large c .

$$\left\{ (G_0, x_0, T'_0) \stackrel{s}{\leftarrow} \mathcal{E}_{0,\lambda} : \text{Enc}_2(1^\lambda, G_0, x_0, T'_0) \right\}$$

$$\left\{ (G_1, x_1, T'_1) \stackrel{s}{\leftarrow} \mathcal{E}_{1,\lambda} : \text{Enc}_2(1^\lambda, G_1, x_1, T'_1) \right\}$$

This concludes the proof. \square

The above composition lemma implies that if there is a sublinear RE with complexity scaling with T^β , one can reduce the complexity by an arbitrary polynomial factor by recursively composing the RE with itself. This fact will be very instrumental later. More precisely,

Lemma 2 (Recursive Composition). *Let α and β be any constants satisfying $0 < \alpha < \beta < 1$. If there is a sublinear RE $(\text{Enc}', \text{Eval}')$ with time complexity*

$$\text{Time}_{\text{Enc}'}(1^\lambda, \Pi, x, T) \leq \text{poly}(\lambda, |\Pi|, |x|)T^\beta,$$

then, there is a sublinear RE $(\text{Enc}, \text{Eval})$ with time complexity

$$\text{Time}_{\text{Enc}}(1^\lambda, \Pi, x, T) \leq \text{poly}(\lambda, |\Pi|, |x|)T^\alpha.$$

Proof of Claim 2. Given any sublinear RE $\text{RE}' = (\text{Enc}', \text{Eval}')$ with time complexity

$$\text{Time}_{\text{Enc}'}(1^\lambda, \Pi, x, T) \leq \text{poly}(\lambda, |\Pi|, |x|)T^\beta$$

We show how to construct a sublinear RE $\text{RE} = (\text{Enc}, \text{Eval})$ with time complexity

$$\text{Time}_{\text{Enc}}(1^\lambda, \Pi, x, T) \leq \text{poly}(\lambda, |\Pi|, |x|)T^\alpha$$

The new scheme RE is constructed by “composing” RE’ iteratively for a sufficiently large, constant, number of times, depending on α and β .

COMPOSING ONCE: Recall that in Section 4.3.4, it was shown that given two ind-secure RE schemes for Turing machines, one can compose them into a new RE scheme that still satisfies indistinguishability security, as shown in Lemma 1. It follows from this lemma that by composing the scheme RE’ with itself, we obtain a new scheme RE₁ as follows.

$$G[\lambda, \Pi, T, s](x) = \text{Enc}'(1^\lambda, \Pi, x, T ; \text{PRG}(s))$$

$$\text{Enc}_1(1^\lambda, \Pi, x, T) : (\hat{G}, \hat{x}) \stackrel{s}{\leftarrow} \text{Enc}'(1^\lambda, G, x, T_G(x))$$

where $T_G(x)$ is an upper bound on the run-time of $G(x)$. We show that RE₁ is more “compact” than RE’ – with a time complexity depending on T^{β^2} as opposed to T^β .

$$T_G(x) = T_{\text{Enc}'}(1^\lambda, \Pi, x, T) \leq \text{poly}(\lambda, |\Pi|, |x|)T^\beta$$

$$T_{\text{Enc}_1}(1^\lambda, \Pi, x, T) = T_{\text{Enc}'}(1^\lambda, G, x, T_G(x)) \leq \text{poly}(\lambda, |\Pi|, |x|)T^{\beta^2}$$

COMPOSING k TIMES: Since RE₁ itself is a sublinear RE scheme, one can apply the same composition technique on it to obtain a new scheme RE₂ whose time complexity depends on $T^{(\beta^2)^2}$. More generally, applying the composition technique recursively for a constant number d of times yields a scheme RE _{d} with time complexity

$$T_{\text{Enc}_d}(1^\lambda, \Pi, x, T) = \text{poly}(\lambda, |\Pi|, |x|)T^{\beta^e}, \quad \text{for } e = 2^d .$$

For a sufficiently large d , $T_{\text{Enc}_d} \leq \text{poly}(\lambda, |\Pi|, |x|)T^\alpha$, which concludes the claim. \square

4.3.5 Simulation Security implies Indistinguishability Security

In this section, we show that simulation security for a randomized encoding scheme implies indistinguishability security for the same scheme. More formally,

Theorem 9. *Let RE be a (λ_0, S) -simulation-secure randomized encoding scheme for a sufficiently large λ_0 . Then RE is also (λ_0, S) -ind-secure.*

Proof. Let Sim be the PPT simulator of $\text{RE} = (\text{Enc}, \text{Eval})$; let c be a sufficiently large constant whose value will become clear in the proof below. To show that RE is (λ_0, S) -ind-secure w.r.t. constant c , consider arbitrary ensembles of distributions $\{D_{0,\lambda}\}$ and $\{D_{1,\lambda}\}$ whose output distributions below are (λ'_0, S') -indistinguishability for $\lambda'_0 \geq \lambda_0$ and $S'(\lambda) \leq S(\lambda)$ for all λ ,

$$\begin{aligned} \mathcal{O}_0 &= \left((\Pi_0, x_0, T_0) \stackrel{\$}{\leftarrow} \mathcal{D}_{0,\lambda} : \Pi_0^{T_0}(x_0), T_0, |\Pi_0|, |x_0| \right) \\ \mathcal{O}_1 &= \left((\Pi_1, x_1, T_1) \stackrel{\$}{\leftarrow} \mathcal{D}_{1,\lambda} : \Pi_1^{T_1}(x_1), T_1, |\Pi_1|, |x_1| \right) \end{aligned}$$

and all parameters $\lambda, |\Pi_b|, |x_b|, T_b \leq B(\lambda)$ for some polynomial B . We show that encoding generated using Enc are (λ'_0, S'') -indistinguishable, where $S''(\lambda) \leq S'(\lambda)/\lambda^c - B(\lambda)^c$.

Consider the following sequence of hybrids:

H_0 :

$$\left((\Pi_0, x_0, T_0) \stackrel{\$}{\leftarrow} \mathcal{D}_{0,\lambda} : \text{Enc}(1^\lambda, \Pi_0, x_0, T_0) \right)$$

H_1 :

$$\left((\Pi_0, x_0, T_0) \stackrel{\$}{\leftarrow} \mathcal{D}_{1,\lambda} : \text{Sim}(1^\lambda, \Pi_0^{T_0}(x_0), 1^{|\Pi_0|}, 1^{|x_0|}, T_0) \right)$$

H_2 :

$$\left((\Pi_1, x_1, T_1) \stackrel{\$}{\leftarrow} \mathcal{D}_{1,\lambda} : \text{Sim}(1^\lambda, \Pi_1^{T_1}(x_1), 1^{|\Pi_1|}, 1^{|x_1|}, T_1) \right)$$

H_3 :

$$\left((\Pi_1, x_1, T_1) \stackrel{\$}{\leftarrow} \mathcal{D}_{1,\lambda} : \text{Enc}(1^\lambda, \Pi_1, x_1, T_1) \right)$$

In other words, our goal is to show that $\{H_0\}$ and $\{H_3\}$ are (λ'_0, S'') -indistinguishable. Towards this, we argue the indistinguishability of every two neighboring hybrids and use a hybrid argument to conclude.

First, it follows directly from the (λ_0, S) -simulation security of RE (and the fact that $\lambda'_0 \geq \lambda_0$) that $\{H_0\}$ and $\{H_1\}$, as well as $\{H_2\}$ and $\{H_3\}$, are $(\lambda'_0, S_{0,1})$ -indistinguishable with $S_{0,1}(\lambda) = S(\lambda) - B(\lambda)^d$ for some constant d .

Next, to argue the indistinguishability of H_1 and H_2 , we first note that the run-time of the simulator $T_{\text{Sim}} = \text{Time}_{\text{Sim}}(1^\lambda, \Pi_b^{T_b}(x_b), 1^{|\Pi_b|}, 1^{|x_b|}, T_b) \leq B(\lambda)^c$ with a sufficiently large constant c determined by Sim, since the length of every input argument is bounded by $B(\lambda)$. Then, it follows from the (λ'_0, S') -indistinguishability of the output distributions $\{O_0\}, \{O_1\}$ that for every adversary of size $S_{1,2}(\lambda) = S'(\lambda) - T_{\text{Sim}} = S'(\lambda) - B(\lambda)^c$, the probability that it distinguishes H_1 and H_2 is bounded by $1/S'(\lambda)$ (as otherwise one can construct a $S'(\lambda)$ -size adversary that distinguishes $\{O_1\}$ and $\{O_2\}$ with probability larger than $1/S'(\lambda)$, by internally running Sim to sample from H_1 or H_2 , and then the distinguisher for H_1 and H_2 to distinguish).

Therefore, it follows from a hybrid argument that for every adversary of size $\min(S_{0,1}(\lambda), S_{1,2}(\lambda))$, the probability of distinguishing H_0 and H_3 is bounded by $2/S_{0,1}(\lambda) + 1/S'(\lambda)$. For sufficiently large $c > d$ and λ'_0 , $\{H_0\}$ and $\{H_3\}$ are (λ'_0, S'') -indistinguishable. \square

4.4 Unbounded-Input IO from Compact RE

In this section, we define our succinct indistinguishability obfuscator for Turing machines. Let $\text{RE} = (\text{Enc}, \text{Eval})$ be a compact randomized encoding scheme for Turing machines with sub-exponential indistinguishability security. Let c be the constant for the security loss associated with the indistinguishability security of RE . We assume without loss of generality that $\text{Enc}(1^\lambda, \cdot, \cdot)$ requires a random tape of length λ . Let PRG be a sub-exponentially secure pseudorandom generator and let ϵ be the constant associated with the sub-exponential security of PRG .

For every $\lambda \in \mathbb{N}$, $D \leq 2^\lambda$, define

$$l(\lambda, -1) = \lambda$$

$$l(\lambda, D) = l(\lambda, D - 1) + (2d\lambda)^{1/\epsilon}$$

where $d > 0$ is any constant strictly greater than c .

Construction 2. Consider a Turing machine Π , security parameter $\lambda \in \mathbb{N}$, and time bound T of Π . For every partial input $s \in \{0, 1\}^*$ with $|s| \leq 2^\lambda$ and $R \in \{0, 1\}^{2l(\lambda, |s|)}$, we recursively define a Turing machine $\tilde{\Pi}_{s,R}$ to be as follows:

When $|s| < 2^\lambda$:

On the empty input, $\tilde{\Pi}_{s,R}$ outputs:

$$\text{Enc}(1^{l(\lambda, |s|+1)}, \tilde{\Pi}_{s_0, R_0}, T'(\lambda, |s| + 1, |\Pi|, \log(T)); R_1)$$

$$\text{Enc}(1^{l(\lambda, |s|+2)}, \tilde{\Pi}_{s_1, R_2}, T'(\lambda, |s| + 1, |\Pi|, \log(T)); R_3)$$

$$\text{Enc}(1^{l(\lambda, |s|+1)}, \Pi, s, T; R_4)$$

where $(R_0, R_1, R_2, R_3, R_4) \leftarrow \text{PRG}(R, 5 \cdot 2l(\lambda, |s| + 1))$ and T' is some fixed polynomial in $\lambda, |s| + 1, |\Pi|$ and $\log(T)$. In the special case when $|s| = 2^\lambda - 1$, the time bound used in the first two encodings is set to T .

On all other inputs, $\tilde{\Pi}_{s,R}$ outputs \perp .

When $|s| = 2^\lambda$:

On the empty input, $\tilde{\Pi}_{s,R}$ outputs $\text{Enc}(1^{l(\lambda, |s| + 1)}, \Pi, s, T; R)$. On all other inputs, $\tilde{\Pi}_{s,R}$ outputs \perp .

We define $T'(\cdot, \cdot, \cdot, \cdot)$ (corresponding to the bound placed on the running time of $\tilde{\Pi}_{s,R}$) to be the smallest polynomial such that for all $\lambda, s \in \{0, 1\}^{\leq 2^\lambda}$, $R \in \{0, 1\}^{2l(\lambda, |s|)}$, Π and T ,

$$\begin{aligned} T'(\lambda, |s|, |\Pi|, \log(T)) &\geq p(\lambda_{|s|+1}, |\tilde{\Pi}_{s,0,R}|, 0, \log(T'_{|s|+1})) \\ &\quad + p(\lambda_{|s|+1}, |\tilde{\Pi}_{s,1,R}|, 0, \log(T'_{|s|+1})) \\ &\quad + p(\lambda_{|s|+1}, |\Pi|, |s|, \log(T)) \\ &\quad + \text{Time}_{\text{PRG}}(R, 5 \cdot 2l(\lambda, |s| + 1)) \end{aligned}$$

where $\lambda_{|s|+1} = l(\lambda, |s| + 1)$, $T'_{|s|+1} = T'(\lambda, |s| + 1, |\Pi|, \log(T))$ (corresponding to the security parameter and time bound used for each of $\tilde{\Pi}_{s,0,R_0}$ and $\tilde{\Pi}_{s,1,R_1}$), Time_{PRG} is the bound on the running time of the PRG, and $p(\cdot, \cdot, \cdot, \cdot)$ is the bound on Time_{Enc} from the compactness of RE . We note that the polynomial T' exists because p is a polynomial, each of $\lambda_{|s|+1}$ and $|\tilde{\Pi}_{s,R}|$ are of size polynomial in $\lambda, |s|$ and $|\Pi|$, and the self-dependence of $T'(\lambda, |s|, |\Pi|, \log(T))$ on $T'_{|s|+1}$ is only poly-logarithmic.

Remark: We note that $|\tilde{\Pi}_{s,R}|$ is always $\text{poly}(\lambda, |\Pi|, |s|, \log(T))$. This is because $\tilde{\Pi}_{s,R}$ is fully described by λ, Π, s, R and T , and the size of each of these is bounded by $\text{poly}(\lambda, |\Pi|, |s|, \log(T))$.

Given this definition of $\tilde{\Pi}_{s,R}$, we define our indistinguishability obfuscator as follows:

Construction 3 (Indistinguishability Obfuscator). *On input $\lambda \in \mathbb{N}$, Turing machine Π and time bound T , define $\tilde{\Pi}$, the indistinguishability obfuscation of Π , to be*

$$\tilde{\Pi} = i\mathcal{O}(1^\lambda, \Pi, T) = \text{Enc}(1^{l(\lambda,0)}, \tilde{\Pi}_{\epsilon,R}, T'(\lambda, 0, |\Pi|, \log(T)))$$

Where ϵ is the empty string, and $R \xleftarrow{\$} \{0, 1\}^{2^{l(\lambda,0)}}$ and T' a fixed polynomial in $\lambda, |\Pi|$ and $\log(T)$, as described above.

Evaluation: The algorithm to evaluate $\tilde{\Pi}$ on input $x \in \{0, 1\}^d, d < 2^\lambda$ proceeds as follows:

1. For every $0 \leq i \leq d$, compute encodings of $\tilde{\Pi}_{x_{\leq i}, R}$ successively, starting with $\tilde{\Pi}$, an encoding of $\tilde{\Pi}_{\epsilon, R}$, and subsequently, for every $0 < i \leq d$, computing the encoding of $\tilde{\Pi}_{x_{\leq i}, R}$ by evaluating the encoding of $\tilde{\Pi}_{x_{< i}, R}$, and selecting the encoding of $\tilde{\Pi}_{x_{\leq i}, R}$ from its output.
2. Evaluate the encoding of $\tilde{\Pi}_{x, R} = \tilde{\Pi}_{x_{\leq d}, R}$ and obtain from its output $(\hat{\Pi}, \hat{x}) = \text{Enc}(1^{l(\lambda, |x|+1)}, \Pi, x, T; R_4)$.
3. Run $\text{Eval}(\hat{\Pi}, \hat{x})$ to obtain $\Pi(x)$.

To analyze the correctness, running time, and compactness of our $i\mathcal{O}$ construction, we make use of the following lemma:

Lemma 3. *Let Π be a polynomial-time TM, $\lambda \in \mathbb{N}$ be a security parameter, and $T \leq 2^\lambda$ be a running time bound. Then, for every $s \in \{0, 1\}^*$ with $|s| \leq 2^\lambda$ and every $R \in \{0, 1\}^{2^{l(\lambda, |s|)}}$, the running time of $\tilde{\Pi}_{s,R}$ is bounded by $T'(\lambda, |s|, |\Pi|, \log(T))$.*

Proof. We prove the lemma by fixing a $\lambda \in \mathbb{N}$, and inducting on the size of s .

Base case: $|s| = 2^\lambda$. In this case, the running time T'_s of $\tilde{\Pi}_{s,R}$ is given by

$$\begin{aligned} T'_s &= \text{Time}_{\text{Enc}}(1^{l(\lambda, |s|+1)}, \Pi, s, T) \\ &\leq p(l(\lambda, |s| + 1), |\Pi|, |s|, \log(T)) \\ &\leq T'(\lambda, |s|, |\Pi|, \log(T)) \end{aligned}$$

This completes the base case.

Inductive step: $|s| < 2^\lambda$ By the induction hypothesis, we assume that the lemma holds for all s' with $|s'| = |s| + 1$, in particular, for $s' = s0$ and $s' = s1$.

Then, an execution of $\tilde{\Pi}_{s,R}$ runs a single evaluation of a PRG, and produces encodings of each of $\tilde{\Pi}_{s0,R_0}$, $\tilde{\Pi}_{s1,R_2}$ and (Π, s) . The PRG runs in time $\text{Time}_{\text{PRG}}(R, 5 \cdot 2l(\lambda, |s|+1))$, while the encoding (Π, s) takes time $\text{Time}_{\text{Enc}}(1^{l(\lambda, |s|+1)}, \Pi, s, T)$. Further, by the inductive hypothesis, the encodings of $\tilde{\Pi}_{s0,R_0}$ and $\tilde{\Pi}_{s1,R_2}$ each take time $\leq T'(\lambda, |s| + 1, |\Pi|, \log(T))$. Combining these facts together, we have that the running time T'_s of $\tilde{\Pi}_{s,R}$ is given by:

$$\begin{aligned} T'_s &= p(\lambda_{|s|+1}, |\tilde{\Pi}_{s0,R}|, 0, \log(T'_{|s|+1})) \\ &\quad + p(\lambda_{|s|+1}, |\tilde{\Pi}_{s1,R}|, 0, \log(T'_{|s|+1})) \\ &\quad + p(\lambda_{|s|+1}, |\Pi|, |s|, \log(T)) \\ &\quad + \text{Time}_{\text{PRG}}(R, 5 \cdot 2l(\lambda, |s| + 1)) \\ &\leq T'(\lambda, |s|, |\Pi|, \log(T)) \end{aligned}$$

This concludes the inductive step, and the lemma follows. \square

Correctness of iO : The correctness of iO applied to any polynomial-length x follows from the correctness of evaluating encodings RE, applied at every level of

the evaluation. More concretely, for each index $i \leq |x|$ of the evaluation, except with probability $\mu(l(\lambda, i))$, the evaluation of $\tilde{\Pi}_{x_{<i}, R}$ correctly produces $\tilde{\Pi}_{x_{\leq i}, R}$. Further, the final evaluation of $\hat{\Pi}, \hat{x}$ produces $\Pi^T(x)$ correctly except with probability $\mu(l(\lambda, i + 1))$.

Crucially, the correctness at each step relies on the fact that each encoding $\tilde{\Pi}_{x_{<i}, R}$ uses time bound $T'(\lambda, i, |\Pi|, \log(T))$, which, as argued above, is sufficiently large to compute $\tilde{\Pi}_{x_{\leq i}, R}$ for the next level.

Overall, the probability of incorrect evaluation is $\leq \sum_{i=1}^{|x|} \mu(\lambda, i)$, which is negligible for any polynomial-length x .

Running time of $i\mathcal{O}$: Again, considering step i , the evaluation at this step takes time $p(l(\lambda, i), |\Pi_{x_{<i}, R}|, 0, T'(\lambda, i, |\Pi|, \log(T)))$, which is $\text{poly}(\lambda, i, |\Pi|, \log(T))$. Further, the evaluation of $\hat{\Pi}, \hat{x}$ is $p(l(\lambda, |x| + 1), |x|, |\Pi|, T)$, which is $\text{poly}(\lambda, |x|, |\Pi|, T)$. Therefore, overall the running time is $\text{poly}(\lambda, |x|, |\Pi|, T)$.

Efficiency of $i\mathcal{O}$: The size of $i\mathcal{O}(1^\lambda, \Pi, T)$ is the same as the size of $\tilde{\Pi}_{\epsilon, R}$, which itself is bounded by $\text{Time}_{\text{Enc}}(1^{l(\lambda, 0)}, \Pi_{\epsilon, R}, T'(\lambda, 0, |\Pi|, \log(T)))$, which is $\text{poly}(\lambda, |\Pi|, \log(T))$ by the efficiency of RE.

4.4.1 Security Proof

Theorem 10. *Let $(\text{Enc}, \text{Eval})$ be a sub-exponentially-indistinguishability-secure, compact randomized encoding scheme and let PRG be a sub-exponentially-secure pseudo-random generator. Then the indistinguishability obfuscator defined in Construction 3 is subexponentially-secure.*

Proof. Consider any pair of ensembles of Turing machines and time bounds $\{\Pi_\lambda^0, \Pi_\lambda^1, T_\lambda\}$ where for every $\lambda \in \mathbb{N}$, $\Pi^0 = \Pi_\lambda^0$, $\Pi^1 = \Pi_\lambda^1$, $T = T_\lambda$,

$$|\Pi^0| = |\Pi^1| \leq \text{poly}(\lambda) \quad |T| \leq \text{poly}(\lambda)$$

$$\forall x, \Pi^{0,T}(x) = \Pi^{1,T}(x)$$

We first introduce some notation to describe the distributions of randomized encodings generated by $i\mathcal{O}(1^\lambda, \Pi_\lambda^0, T_\lambda)$ and $i\mathcal{O}(1^\lambda, \Pi_\lambda^1, T_\lambda)$. For $\lambda \in \mathbb{N}$, $s \in \{0, 1\}^*$, $|s| \leq 2^\lambda$, we define the following distributions

$$D_{\lambda,0,s} = \text{Enc}(1^{l(\lambda,|s|)}, \tilde{\Pi}_{s,R}^0, T')$$

$$D_{\lambda,1,s} = \text{Enc}(1^{l(\lambda,|s|)}, \tilde{\Pi}_{s,R}^1, T')$$

where R is uniformly random, T' is as described in Construction 2 and $\tilde{\Pi}_{s,R}^b$ is defined for the Turing machine Π_λ^b , security parameter λ and time bound T_λ . We will show something stronger than the theorem statement. In particular, we have the following claim.

Claim 1. *There exists $\lambda_0, \epsilon \in \mathbb{N}$ such that for every $\lambda > \lambda_0$, for every $s \in \{0, 1\}^*$, $|s| \leq 2^\lambda$ we have that the distributions $D_{\lambda,0,s}$ and $D_{\lambda,1,s}$ are $S(\lambda)$ indistinguishable where $S(\lambda) \geq 10 \cdot 2^{l(\lambda,|s|-1)^\epsilon}$.*

Using the above claim with s as the empty string and recalling $l(\lambda, 0) = \lambda$, the theorem statement follows. Therefore, in the remainder of the proof, we prove the above claim.

Proof of claim Let ϵ be the larger of the constants associated with the sub-exponential security of the pseudorandom generator PRG and the indistinguishability security of the encoding scheme (Enc, Eval) (these constants are also named ϵ in their respective security definitions). Similarly, We consider λ_0 to be

large enough so that the security of the encoding scheme (Enc, Eval) and the pseudorandom generator PRG is applicable. We will actually require a larger λ_0 so that certain asymptotic conditions (depending only on the polynomial size bounds of Π_λ^0 , Π_λ^1 and T_λ) hold, which we make explicit in the remainder of the proof. For every $\lambda > \lambda_0$, we prove the claim by induction on $|s|$. Our base case will be when $|s| = 2^\lambda$ and in the inductive step we show the claim holds for all s of a particular length d , if it holds for all s of length $d + 1$.

Induction statement, for a fixed $\lambda > \lambda_0$: For every $s \in \{0, 1\}^{\leq 2^\lambda}$, the distributions $D_{\lambda,0,s}$ and $D_{\lambda,1,s}$ are $10 \cdot 2^{l(\lambda,|s|-1)\epsilon}$ indistinguishable.

Base case: $|s| = 2^\lambda$.

In this case, recall that the output of $\tilde{\Pi}_{s,R}^b$ is simply $(\hat{\Pi}_\lambda^{b,T}, \hat{s})$. We first claim that, for all s , $(\hat{\Pi}_\lambda^{0,T}, \hat{s})$ and $(\hat{\Pi}_\lambda^{1,T}, \hat{s})$ are $2^{\lambda'\epsilon}$ indistinguishable where $\lambda' = l(\lambda, |s|)$, as follows.

Recall that the output of evaluating $\hat{\Pi}_\lambda^{b,T}, \hat{s}$ is simply $\Pi_\lambda^{b,T}(s)$. Since we have that $\Pi_\lambda^{0,T}(s) = \Pi_\lambda^{1,T}(s)$ for all s , we can apply the security of the randomized encoding scheme. More concretely, since the output (point) distributions are identical, they are $10 \cdot 2^{\lambda'\epsilon}$ -indistinguishable where $\lambda' = l(\lambda, |s| + 1)$. Let $B(\cdot)$ be a polynomial such that $B(\lambda')$ bounds from above $|\Pi^b|, |s|$ and T . By the security of the encoding scheme, the encodings $(\hat{\Pi}_\lambda^{0,T}, \hat{s})$ and $(\hat{\Pi}_\lambda^{1,T}, \hat{s})$ are S' indistinguishable where

$$S' \geq \frac{10 \cdot 2^{l(\lambda,|s|+1)\epsilon}}{l(\lambda, |s| + 1)^c} - B(l(\lambda, |s| + 1))^c \geq \frac{10 \cdot 2^{l(\lambda,|s|+1)\epsilon}}{l(\lambda, |s| + 1)^d} \geq 10 \cdot 2^{l(\lambda,|s|)\epsilon}$$

where the first inequality holds for sufficiently large λ and in the second inequality, we use the fact that $l(\lambda, |s| + 1) = l(\lambda, |s|) + \lambda^{d/\epsilon}$. Thus $(\hat{\Pi}_\lambda^{0,T}, \hat{s})$ and $(\hat{\Pi}_\lambda^{1,T}, \hat{s})$ are $10 \cdot 2^{l(\lambda,|s|)\epsilon}$ -indistinguishable.

Now, recall that the output of $\tilde{\Pi}_{s,R}^b$ is simply $(\hat{\Pi}_\lambda^{b,T}, \hat{s})$. By the above argument, we have that, for all s , $(\hat{\Pi}_\lambda^{0,T}, \hat{s})$ and $(\hat{\Pi}_\lambda^{1,T}, \hat{s})$ are 2^{λ^ϵ} -indistinguishable where $\lambda' = l(\lambda, |s|)$. Let B' be the polynomial such that $B'(l(\lambda, |s|))$ bounds $|\tilde{\Pi}_{s,R}^b|$ and the running time of $\tilde{\Pi}_{s,R}^b$ as per Lemma 3. The encodings $D_{\lambda,0,s}$ and $D_{\lambda,1,s}$ are S' indistinguishable where

$$S' \geq \frac{10 \cdot 2^{l(\lambda, |s|)^\epsilon}}{l(\lambda, |s|)^c} - B'(l(\lambda, |s|))^c \geq \frac{10 \cdot 2^{l(\lambda, |s|+1)^\epsilon}}{l(\lambda, |s|)^d} \geq 10 \cdot 2^{l(\lambda, |s|-1)^\epsilon}$$

where, as before, the first inequality holds for sufficiently large λ and in the second inequality, we use the fact that $l(\lambda, |s|+1) = l(\lambda, |s|) + \lambda^{d/\epsilon}$. Hence the claim holds for $|s| = 2^\lambda$.

Inductive step: $|s| < 2^\lambda$. By the induction hypothesis, we assume the claim holds for all s' such that $|s'| = |s| + 1$. Recall that the output of $\tilde{\Pi}_{s,R}^b$ (where $R \stackrel{\$}{\leftarrow} \{0, 1\}^{2l(\lambda, |s|)}$) is

$$\text{Enc}(1^{l(\lambda, |s|+1)}, \tilde{\Pi}_{s_0, R_0}^b, T'; R_1)$$

$$\text{Enc}(1^{l(\lambda, |s|+1)}, \tilde{\Pi}_{s_1, R_2}^b, T'; R_3)$$

$$\text{Enc}(1^{l(\lambda, |s|+1)}, \Pi_\lambda^b, s, T; R_4)$$

where $(R_0, R_1, R_2, R_3, R_4) \leftarrow \text{PRG}(R, 5 \cdot 2l(\lambda, |s| + 1))$. Let H^b denote the above output distribution. We will show H^0 and H^1 are indistinguishable by a hybrid argument as follows.

- Let G_1 be a hybrid distribution exactly as H^0 except that $(R_0, R_1, R_2, R_3, R_4) \stackrel{\$}{\leftarrow} \{0, 1\}^{5 \cdot 2l(\lambda, |s|+1)}$. We claim that for both the distributions H^0 and G_1 are $5 \cdot 2^{\lambda^\epsilon}$ indistinguishable where $\lambda' = l(\lambda, |s|)$.

This follows from the PRG security as follows: any size $5 \cdot 2^{\lambda^\epsilon}$ adversary A that distinguishes H^0 and G_1 can be turned into an adversary A' that can

break the PRG security with seed length $2\lambda'$ with the same advantage. A' has $\Pi_\lambda^0, \Pi_{\lambda'}^1, T_\lambda$ and s hardcoded in it. Hence, the size of A' is

$$5 \cdot 2^{\lambda'\epsilon} + \text{poly}(\lambda) + \text{poly}(|s|) \leq 5 \cdot 2^{\lambda'\epsilon} + \text{poly}(\lambda') \leq 2^{(2\lambda')\epsilon}$$

where the last inequality holds when λ is sufficiently large. Hence, A' breaks the $2^{(2\lambda')\epsilon}$ -security of PRG and we have a contradiction.

Writing out the components of G_1 , we have that it is identical to

$$G_1 \equiv D_{\lambda,0,s,0}, D_{\lambda,0,s,1}, \text{Enc}(1^{l(\lambda,|s|+1)}, \Pi_\lambda^0, s, T_\lambda; R)$$

- Let G_2 be a hybrid distribution obtained by modifying the first component of G_1 as follows.

$$G_2 \equiv D_{\lambda,1,s,0}, D_{\lambda,0,s,1}, \text{Enc}(1^{l(\lambda,|s|+1)}, \Pi_\lambda^0, s, T_\lambda; R)$$

We show that G_1 and G_2 are $5 \cdot 2^{\lambda'\epsilon}$ indistinguishable. This follows from the induction hypothesis as follows: any size $5 \cdot 2^{\lambda'\epsilon}$ adversary A that distinguishes G_1 and G_2 with advantage better than $1/(5 \cdot 2^{\lambda'\epsilon})$ can be turned into an adversary A' that can distinguish $D_{\lambda,0,s,0}$ and $D_{\lambda,1,s,0}$ with the same advantage. As before, A' has $\Pi_\lambda^0, \Pi_{\lambda'}^1, T_\lambda$ and s hardcoded in it, and therefore the size of A' is at most $5 \cdot 2^{\lambda'\epsilon} + \text{poly}(\lambda') \leq 10 \cdot 2^{\lambda'\epsilon}$. Hence, A' breaks the induction hypothesis that says $D_{\lambda,0,s,0}$ and $D_{\lambda,1,s,0}$ are $10 \cdot 2^{\lambda'\epsilon}$ -indistinguishable.

- Similarly, let G_3 be a hybrid distribution obtained by modifying the second component of G_2 as follows.

$$G_3 \equiv D_{\lambda,1,s,0}, D_{\lambda,1,s,1}, \text{Enc}(1^{l(\lambda,|s|+1)}, \Pi_\lambda^0, s, T_\lambda; R)$$

Similarly as above, we have that G_2 and G_3 are $5 \cdot 2^{\lambda'\epsilon}$ -indistinguishable.

- Let G_4 be a hybrid distribution obtained by modifying the third component of G_3 as follows.

$$G_4 \equiv D_{\lambda,1,s,0}, D_{\lambda,1,s,1}, \text{Enc}(1^{l(\lambda,|s|+1)}, \Pi_{\lambda}^1, s, T_{\lambda}; R)$$

We show G_3 and G_4 are $5 \cdot 2^{\lambda^\epsilon}$ -indistinguishable. First, since $\Pi_{\lambda}^{0,T}(s) = \Pi_{\lambda}^{1,T}(s)$, by the security of the encoding scheme, we have that the encodings that form the third component of G_3 and G_4 are S' indistinguishable where, similar to the base case, $B(l(\lambda, |s|))$ bounds from above $|\Pi_{\lambda}^b|$, $|s|$ and T

$$S' \geq \frac{10 \cdot 2^{l(\lambda,|s|)^\epsilon}}{l(\lambda, |s|)^c} - B(l(\lambda, |s|))^c \geq \frac{10 \cdot 2^{l(\lambda,|s|)^\epsilon}}{l(\lambda, |s|)^d} \geq 10 \cdot 2^{l(\lambda,|s|-1)^\epsilon}$$

Hence by a similar argument as before, the hybrid distributions are $5 \cdot 2^{\lambda^\epsilon}$ -indistinguishable.

- Finally we observe that G_4 and H^1 are $5 \cdot 2^{\lambda^\epsilon}$ -indistinguishable just as G_1 and H^0 were. By a simple hybrid argument, we have that H^0 and H^1 are 2^{λ^ϵ} -indistinguishable.

Recall that H^0 and H^1 are the distributions of outputs of $\tilde{\Pi}_{s,R}^0$ and $\tilde{\Pi}_{s,R}^1$ respectively. By the security of the randomized encoding scheme, the encodings of these machines, *i.e.* $D_{\lambda,0,s}$ and $D_{\lambda,1,s}$ are $S'(\lambda)$ -indistinguishable where

$$S'(\lambda) \geq \frac{2^{l(\lambda,|s|)^\epsilon}}{l(\lambda, |s|)^c} - B'(l(\lambda, |s|))^c \geq \frac{2^{l(\lambda,|s|)^\epsilon}}{l(\lambda, |s|)^d} \geq \frac{2^{l(\lambda,|s|-1)^\epsilon} \cdot 2^{(2d\lambda)}}{2^{d\lambda} \cdot (2d\lambda)^{d/\epsilon}} \geq 10 \cdot 2^{l(\lambda,|s|-1)^\epsilon}$$

where $B'(l(\lambda, |s|))$ bounds from above $|\Pi_{s,R}^b|$ and T' as in Lemma 3. The second inequality holds for sufficiently large λ . In the third inequality, we use the fact that $l(\lambda, |s|) \leq |s|(2d\lambda)^{1/\epsilon} \leq 2^\lambda(2d\lambda)^{1/\epsilon}$ and the last inequality holds for sufficiently large λ .

□

4.4.2 Nice Distributions

Later in Section 4.7, we show that compact RE does not exist for general distributions in the plain model. However, here we observe that the above construction of unbounded input IO relies only on compact RE for certain “special purpose” distributions that is not ruled out by the impossibility result in Section 4.7. We now abstract out the structure of these special purpose distributions. Let $\text{RE} = (\text{Enc}, \text{Dec})$ be a randomized encoding scheme; we define “nice” distributions w.r.t. RE.

0-nice distributions: We say that a pair of distribution ensembles $\{\mathcal{D}_{0,\lambda}\}$ and $\{\mathcal{D}_{1,\lambda}\}$ are *0-nice* if $D_{0,\lambda}$ always outputs a fixed tuple (Π_0, x, T) while $D_{1,\lambda}$ always outputs a fixed tuple (Π_1, x, T) , satisfying that $\Pi_0^T(x) = \Pi_1^T(x)$.

k -nice distributions: We say that a pair of distribution ensembles $\{\mathcal{D}_{0,\lambda}\}$ and $\{\mathcal{D}_{1,\lambda}\}$ are *k -nice* if there exist some $\ell = \text{poly}(\lambda)$ pairs of distributions $(\{\mathcal{E}_{0,\lambda}^i\}, \{\mathcal{E}_{1,\lambda}^i\})_{i \in [\ell]}$, where the i^{th} pair is k^i -nice with $k^i \leq k - 1$, such that, $\mathcal{D}_{b,\lambda}$ samples tuple (Π_b, x_b, T_b) satisfying the following:

- For each $i \in [\ell]$, sample $(\Lambda_b^i, z_b^i, T_b^i) \stackrel{\$}{\leftarrow} \mathcal{E}_{b,\lambda}^i$.
- The output of $\Pi_b(x_b)$ consists of ℓ randomized encodings, where the i^{th} encoding is in the support of $\text{Enc}(1^{\lambda'}, \Lambda_b^i, z_b^i, T_b^i)$, for some $\lambda' = \text{poly}(\lambda)$.

Finally, we say that a pair of distribution ensembles $\{\mathcal{D}_{0,\lambda}\}$ and $\{\mathcal{D}_{1,\lambda}\}$ are *nice* w.r.t. RE if they are *k -nice* w.r.t. RE for some integer k .

Our construction of unbounded input IO and its analysis in previous sections relies only on compact RE for nice distribution ensembles. Hence we can refine Theorem 10 to the following:

Proposition 1. *Assume the existence of a compact randomized encoding scheme RE which is sub-exponentially-indistinguishability-secure for every pair of distribution ensemble that are nice w.r.t. RE; assume further the existence of sub-exponentially secure one-way functions. Then, there is an unbounded-input indistinguishability obfuscator for Turing machines.*

We stress again that compact RE for nice distributions is not ruled out by the impossibility result in Section 4.7. Hence, we obtain unbounded input IO from a new assumption different from the extractability assumptions used in previous work [32, 1, 85].

Candidate Construction: Finally, we describe a candidate construction of compact RE for nice distributions using the KLV indistinguishability obfuscator for bounded-input Boolean Turing machines: Given input $(1^\lambda, M, x, T)$, the encoding is an obfuscation, using the KLV scheme, of the program $\Pi_{M,x}$ that on input $i \in [T]$ outputs the i^{th} bit of the output $M^T(x)$. Since $\Pi_{M,x}$ is Boolean, the KLV obfuscator can be applied, and the encoding time is $\text{poly}(\lambda, |M|, |x|, \log T)$ (hence compact). By the security of indistinguishability obfuscation, for any M_1, x_1 and M_2, x_2 with identical outputs, their encodings are indistinguishable, and thus this construction is a weak compact RE. We here consider it also a candidate construction for compact RE with distributional indistinguishability.

4.5 Bounded-Input IO from Sublinear RE

In this section, we construct a bounded-input indistinguishability obfuscator for Turing machines, using randomized encoding schemes with sublinear com-

compactness. The construction is very similar to the construction described earlier in Section 4.4. The main difference is that due to the fact that we only use sub-linear compactness, the depth of the tree of encodings we construct must be bounded to some polynomial size given as a parameter, rather than being of depth 2^λ . Further, the correctness and efficiency analysis is slightly different to account for the semi-compactness of the RE scheme used, as opposed to full compactness in the previous construction.

Let $\text{RE} = (\text{Enc}, \text{Eval})$ be a randomized encoding scheme for Turing machines with sub-exponential indistinguishability security and sublinear efficiency. Let c be the constant associated with the security loss in the security of $(\text{Enc}, \text{Eval})$. We assume that $\text{Enc}(1^\lambda, \cdot, \cdot)$ requires a random tape of length λ , and this is without loss of generality for two reasons: First, one can always apply a PRG to expand the λ -bit random string to a pseudo-random string of arbitrary polynomial length, and second, by Claim 2 in Section 4.3.4 it is without loss of generality to assume that we start with a RE scheme with a sufficiently small sublinear time complexity and thus even counting the time for PRG expansion, the overall time complexity is still sublinear. Let PRG be a sub-exponentially secure pseudorandom generator and let ϵ be the constant associated with the sub-exponential security of PRG.

For every $\lambda \in \mathbb{N}$, $D \leq 2^\lambda$, define

$$l(\lambda, -1) = \lambda$$

$$l(\lambda, D) = l(\lambda, D - 1) + (2d\lambda)^{1/\epsilon}$$

where d is any constant strictly greater than c .

Construction 4. Consider a Turing machine Π , security parameter $\lambda \in \mathbb{N}$, an input-

length bound n , and time bound T on the running time of Π . For every partial input $s \in \{0, 1\}^*$ with $|s| \leq n$ and $R \in \{0, 1\}^{2l(\lambda, |s|)}$, we recursively define a Turing machine $\tilde{\Pi}_{s,R}$ as follows:

When $|s| < n$:

On the empty input, $\tilde{\Pi}_{s,R}$ outputs:

$$\text{Enc}(1^{l(\lambda, |s|+1)}, \tilde{\Pi}_{s0, R_0}, T'(\lambda, |s| + 1, |\Pi|, n, T); R_1)$$

$$\text{Enc}(1^{l(\lambda, |s|+1)}, \tilde{\Pi}_{s1, R_2}, T'(\lambda, |s| + 1, |\Pi|, n, T); R_3)$$

$$\text{Enc}(1^{l(\lambda, |s|+1)}, \Pi, s, T; R_4)$$

where $(R_0, R_1, R_2, R_3, R_4) \leftarrow \text{PRG}(R, 5 \cdot 2l(\lambda, |s| + 1))$ and T' is a fixed polynomial in $\lambda, |s| + 1, |\Pi|, n$ and T , defined below. In the special case when $|s| = n - 1$, the time bound used in the first two encodings is set to T .

On all other inputs, $\tilde{\Pi}_{s,R}$ outputs \perp .

When $|s| = n$:

On the empty input, $\tilde{\Pi}_{s,R}$ outputs $\text{Enc}(1^{l(\lambda, |s|+1)}, \Pi, s, T; R)$. On all other inputs, $\tilde{\Pi}_{s,R}$ outputs \perp .

Let $p(\cdot)$ and ϵ' respectively be the polynomial and constant corresponding to the sub-linear efficiency of RE. For any λ, n, Π and T , we define the following terms (which are implicitly functions of λ, n, Π and T):

$$\lambda_n = l(\lambda, n)$$

$$\lambda_{n+1} = l(\lambda, n + 1)$$

$$\tilde{\Pi}_n = \tilde{\Pi}_{0^n, 0^{2\lambda_n}}$$

$$A = 2 \cdot p(1^{\lambda_n}, |\tilde{\Pi}_n|, 0)$$

$$B = p(1^{\lambda_{n+1}}, |\Pi|, n)T^{1-\epsilon'}$$

$$C = \text{Time}_{\text{PRG}}(0^{2\lambda_n}, 5 \cdot 2\lambda_{n+1})$$

We note that $\tilde{\Pi}_n$ is the largest size of any $\tilde{\Pi}_{s,R}$ we ever need to consider, and further, that it has a description of polynomial size, and hence every $\tilde{\Pi}_{s,R}$ has polynomial size.

We define the polynomial $T'(\cdot, \cdot, \cdot, \cdot)$ (corresponding to the bound placed on the running time of $\tilde{\Pi}_{s,R}$), for all $\lambda, s \in \{0, 1\}^{\leq n}, R \in \{0, 1\}^{2l(\lambda, |s|)}, \Pi$ and T , to be

$$T'(\lambda, |s|, |\Pi|, n, T) = (n - |s| + 1) \cdot (A^{1/\epsilon'} + B + C)$$

where each of A, B and C are defined relative to $\lambda, |\Pi|, n$ and T given as input to T' .

Given this definition of $\tilde{\Pi}_{s,R}$ and T' , we define our indistinguishability obfuscator as follows:

Construction 5 (Indistinguishability Obfuscator). *On input $\lambda \in \mathbb{N}$, Turing machine Π , input length bound n and time bound T , define $\tilde{\Pi}$, the indistinguishability obfuscation of Π , to be*

$$\tilde{\Pi} = i\mathcal{O}(1^\lambda, \Pi, n, T) = \text{Enc}(1^{l(\lambda, 0)}, \tilde{\Pi}_{\epsilon, R}, T'(\lambda, 0, |\Pi|, n, T))$$

Where ϵ is the empty string, and $R \xleftarrow{\$} \{0, 1\}^{2l(\lambda, 0)}$ and T' a fixed polynomial in $\lambda, |\Pi|, n$ and T , as described above.

Evaluation: The algorithm to evaluate $\tilde{\Pi}$ on input $x \in \{0, 1\}^d, d \leq n$ proceeds as follows:

1. For every $0 \leq i \leq d$, compute encodings of $\tilde{\Pi}_{x_{\leq i}, R}$ successively, starting with $\tilde{\Pi}$, an encoding of $\tilde{\Pi}_{\epsilon, R}$, and subsequently, for every $0 < i \leq d$, computing the encoding of $\tilde{\Pi}_{x_{\leq i}, R}$ by evaluating the encoding of $\tilde{\Pi}_{x_{< i}, R}$, and selecting the encoding of $\tilde{\Pi}_{x_{\leq i}, R}$ from its output.
2. Evaluate the encoding of $\tilde{\Pi}_{x, R} = \tilde{\Pi}_{x_{\leq d}, R}$ and obtain from its output $(\hat{\Pi}, \hat{x}) = \text{Enc}(1^{l(\lambda, |x|+1)}, \Pi, x, T)$.
3. Run $\text{Eval}(\hat{\Pi}, \hat{x})$ to obtain $\Pi(x)$.

To analyze the correctness, running time, and compactness of our iO construction, we make use of the following lemma:

Lemma 4. *Let Π be a polynomial-time TM, $\lambda \in \mathbb{N}$ be a security parameter, n an input length bound, and $T \leq 2^\lambda$ be a running time bound. Then, for every $s \in \{0, 1\}^*$ with $|s| \leq 2^\lambda$ and every $R \in \{0, 1\}^{2^{l(\lambda, |s|)}}$, the running time of $\tilde{\Pi}_{s, R}$ is bounded by $T'(\lambda, |s|, |\Pi|, n, T)$.*

Proof. We prove the lemma by fixing $\lambda \in \mathbb{N}$, and inducting on the size of s .

Base case: $|s| = n$. In this case, the running time T'_s of $\tilde{\Pi}_{s, R}$ is given by

$$\begin{aligned}
T'_s &= \text{Time}_{\text{Enc}}(1^{l(\lambda, |s|+1)}, \Pi, s, T) \\
&\leq p(l(\lambda, |s| + 1), |\Pi|, n)T^{1-\epsilon'} \\
&\leq B \\
&\leq T'(\lambda, |s|, |\Pi|, n, T)
\end{aligned}$$

This completes the base case.

Inductive step: $|s| < n$ By the induction hypothesis, we assume that the lemma holds for all s' with $|s'| = |s| + 1$, in particular, for $s' = s0$ and $s' = s1$.

Then, an execution of $\tilde{\Pi}_{s,R}$ runs a single evaluation of a PRG, and produces encodings of each of $\tilde{\Pi}_{s_0,R_0}$, $\tilde{\Pi}_{s_1,R_2}$ and (Π, s) . The PRG runs in time $\text{Time}_{\text{PRG}}(R, 5 \cdot 2^{\lambda(\lambda, |s| + 1)}) \leq C$, while the encoding (Π, s) takes time $\text{Time}_{\text{Enc}}(1^{\lambda(\lambda, |s| + 1)}, \Pi, s, T) \leq B$. By the inductive hypothesis, the machines $\tilde{\Pi}_{s_0,R_0}$ and $\tilde{\Pi}_{s_1,R_2}$ each run in time $\leq T'(\lambda, |s| + 1, |\Pi|, T)$. Then, we have that the encoding time of $\tilde{\Pi}_{s_0,R_0}$ and $\tilde{\Pi}_{s_1,R_2}$ is bounded by:

$$\begin{aligned}
& 2 \cdot \text{Time}_{\text{Enc}}(1^{\lambda(\lambda, |s| + 1)}, \tilde{\Pi}_{s_0,R_0}, 0, T'(\lambda, |s| + 1, |\Pi|, T)) \\
& \leq 2 \cdot p(1^{\lambda(\lambda, |s| + 1)}, \tilde{\Pi}_{s_0,R_0}, 0) \cdot T'(\lambda, |s| + 1, |\Pi|, T)^{1-\epsilon'} \\
& \leq A \cdot T'(\lambda, |s| + 1, |\Pi|, T)^{1-\epsilon'} \\
& \leq A \cdot T'(\lambda, |s| + 1, |\Pi|, T)^1 \cdot T'(\lambda, |s| + 1, |\Pi|, T)^{-\epsilon'} \\
& \leq A \cdot T'(\lambda, |s| + 1, |\Pi|, T) \cdot (A^{1/\epsilon'})^{-\epsilon} \\
& \leq T'(\lambda, |s| + 1, |\Pi|, T)
\end{aligned}$$

Combining these facts together, we have that the running time T'_s of $\tilde{\Pi}_{s,R}$ is given by:

$$\begin{aligned}
T'_s & \leq T'(\lambda, |s| + 1, |\Pi|, T) + B + C \\
& \leq (n - (|s| + 1) + 1) \cdot (A^{1/\epsilon'} + B + C) + B + C \\
& \leq (n - |s| + 1) \cdot (A^{1/\epsilon'} + B + C) \\
& \leq T'(\lambda, |s|, |\Pi|, T)
\end{aligned}$$

This concludes the inductive step, and the lemma follows. \square

Correctness of iO : As in the construction in Section 4.4, the correctness of iO applied to any polynomial-length x follows from the correctness of evaluating encodings RE, applied at every level of the evaluation. More concretely, for each

index $i \leq |x|$ of the evaluation, except with probability $\mu(l(\lambda, i))$, the evaluation of $\tilde{\Pi}_{x_{<i}, R}$ correctly produces $\tilde{\Pi}_{x_{\leq i}, R}$. Further, the final evaluation of $\hat{\Pi}, \hat{x}$ produces $\Pi^T(x)$ correctly except with probability $\mu(l(\lambda, i + 1))$.

Crucially, the correctness at each step relies on the fact that each encoding $\tilde{\Pi}_{x_{<i}, R}$ uses time bound $T'(\lambda, i, |\Pi|, \log(T))$, which, as argued above, is sufficiently large to compute $\tilde{\Pi}_{x_{\leq i}, R}$ for the next level.

Overall, the probability of incorrect evaluation is $\leq \sum_{i=1}^{|x|} \mu(\lambda, i)$, which is negligible for any polynomial-length x .

Running time of $i\mathcal{O}$: Again, considering step i , the evaluation at this step takes time $p(l(\lambda, i), |\Pi_{x_{<i}, R}|, 0, T'(\lambda, i, |\Pi|, T))$, which is $\text{poly}(\lambda, i, |\Pi|, T)$. Further, the evaluation of $\hat{\Pi}, \hat{x}$ to produce the final output of the $i\mathcal{O}$ is $p(l(\lambda, |x| + 1), |x|, |\Pi|, T)$, which is $\text{poly}(\lambda, |x|, |\Pi|, T)$. Therefore, overall the running time is $\text{poly}(\lambda, |x|, |\Pi|, T)$.

Efficiency of $i\mathcal{O}$: The size of $i\mathcal{O}(1^\lambda, \Pi, T)$ is the same as the size of $\tilde{\Pi}_{\epsilon, R}$, which itself is bounded by $\text{Time}_{\text{Enc}}(1^{l(\lambda, 0)}, \Pi_{\epsilon, R}, T'(\lambda, 0, |\Pi|, T))$, which is $\text{poly}(\lambda, |\Pi|, T)$ by the efficiency of RE.

Security of $i\mathcal{O}$: We note that the security proof for Construction 3 presented in Section 4.4 carries over exactly to the construction presented in this section. The only difference is that the base case for the induction starts from $|s| = n$ rather than $|s| = 2^\lambda$. Given this change, exactly the same inductive argument can be used to show that subexponential security of RE implies subexponential security of the $i\mathcal{O}$ construction given above.

4.6 Bounded-Input IO from Compact RE in the CRS Model

In this section we consider compact RE schemes for Turing machines in the *common reference string* (CRS) model. We show that (1) such encoding schemes can be constructed from compact functional encryption for circuits, and that (2) such encoding schemes suffice to get IO for circuits, which then by [90] suffices to get bounded-input IO for Turing machines.

4.6.1 Randomized Encoding Schemes in the CRS model

We first formally define a randomized encoding scheme for a class of Turing machines in the CRS model. In this model, a one-time setup is performed which takes (in addition to the security parameter) a bound on machine size, input length, running time and output length. Only computations that respect these bounds can be encoded using this setup. The setup outputs a *long* CRS (the length is polynomial in the aforementioned bounds) and a *short* public encoding key (which depends only on the security parameter). The public encoding key is used by the encoding algorithm, which produces encodings that are *compact* as before. The CRS is used by the evaluation algorithm.

Definition 20 (Randomized Encoding Schemes in the CRS Model). *A Randomized Encoding scheme RE for a class of Turing machines $\{\mathcal{M}_\lambda\}$ in the CRS model consists of the following algorithms:*

- $(\text{crs}, pk) \stackrel{s}{\leftarrow} \text{Setup}(1^\lambda, 1^m, 1^n, 1^T, 1^l)$: Setup gets as input (in unary) the security parameter λ , a machine size bound m , input length bound n , time bound T and output length bound l .

- $\hat{\Pi}_x \stackrel{\$}{\leftarrow} \text{Enc}(pk, \Pi, x)$: Enc is probabilistic and gets as input a public key pk generated by Setup, Turing machine $\Pi \in \mathcal{M}_\lambda$ and input x . It outputs an encoding $\hat{\Pi}_x$ ⁷.
- $y \leftarrow \text{Eval}(\hat{\Pi}_x, \text{crs})$: On input $\hat{\Pi}_x$ produced by Enc and crs produced by Setup, Eval outputs y .

Correctness: For every security parameters $\lambda \in \mathbb{N}$, $m, n, T, l \in \mathbb{N}$, Turing machine $\Pi \in \mathcal{M}_\lambda$ and input x , such that, $|\Pi| \leq m$, $|x| \leq n$, and $|\Pi^T(x)| \leq l$, we have that

$$\Pr \left[\begin{array}{l} (\text{crs}, pk) \stackrel{\$}{\leftarrow} \text{Setup}(1^\lambda, 1^m, 1^n, 1^T, 1^l) \\ \hat{\Pi}_x \stackrel{\$}{\leftarrow} \text{Enc}(pk, \Pi, x) \end{array} : \text{Eval}(\hat{\Pi}_x, \text{crs}) = \Pi^T(x) \right] = 1$$

The simulation security in the CRS model is essentially the same as that in the plain model (Definition 15), except that simulator can additionally simulate the CRS.

Definition 21. A randomized encoding scheme RE for a class of Turing machines $\{\mathcal{M}_\lambda\}$ in the CRS model satisfies $(\lambda_0, S(\cdot))$ -**simulation security**, if there exists a PPT algorithm Sim and a constant c , such that, for every ensemble $\{\Pi_\lambda, x_\lambda, m_\lambda, n_\lambda, l_\lambda, T_\lambda\}$ where $\Pi_\lambda \in \mathcal{M}_\lambda$ and $|\Pi_\lambda|, |x_\lambda|, m_\lambda, n_\lambda, l_\lambda, T_\lambda \leq B(\lambda)$ for some polynomial B , the following ensembles are $(\lambda_0, S'(\lambda))$ indistinguishable, with $S'(\lambda) = S(\lambda) - B(\lambda)^c$ for all $\lambda \in N$.

$$\left\{ (\text{crs}, pk) \stackrel{\$}{\leftarrow} \text{Setup}(1^\lambda, 1^m, 1^n, 1^T, 1^l), \hat{\Pi}_x \stackrel{\$}{\leftarrow} \text{Enc}(pk, \Pi, x) : (\text{crs}, pk, \hat{\Pi}_x) \right\} \\ \left\{ (\text{crs}, pk, \hat{\Pi}_x) \stackrel{\$}{\leftarrow} \text{Sim}(1^\lambda, \Pi^T(x), 1^{|\Pi|}, 1^{|x|}, 1^m, 1^n, 1^T, 1^l) : (\text{crs}, pk, \hat{\Pi}_x) \right\}$$

where subscripts of security parameter are suppressed.

⁷Encoding $\hat{\Pi}_x$ can be viewed as the combination of the program encoding $\hat{\Pi}$ and the input encoding \hat{x} of Definition 14

Indistinguishability-security in the CRS model can be defined similar to that in the plain model, except now we need to work with distributions $D_{b,\lambda}$ that samples $(\Pi_b, x_b, T_b, m_b, n_b, l_b)$.

Definition 22 (Distributional $(\lambda_0, S(\cdot))$ -Indistinguishability Security). *A randomized encoding scheme RE for a class of Turing machines $\{\mathcal{M}_\lambda\}$ in the CRS model satisfies $(\lambda_0, S(\cdot))$ -ind-security, if the following is true w.r.t. some constant $c > 0$: For every ensembles of distributions $\{D_{0,\lambda}\}$ and $\{D_{1,\lambda}\}$ with the following property:*

1. *there exists a polynomial B , such that, for every $b \in \{0, 1\}$, $D_{b,\lambda}$ is a distribution over tuples of the form $(\Pi_b, x_b, T_b, m_b, n_b, l_b)$ with polynomial size and $\lambda, |\Pi_b|, |x_b|, T_b, m_b, n_b, l_b \leq B(\lambda)$.*
2. *there exist an integer $\lambda'_0 \geq \lambda_0$, and a function S' with $\leq S'(\lambda) \leq S(\lambda)$ for all λ , such that, the ensembles of output distributions $\{\mathcal{O}_{0,\lambda}\}$ and $\{\mathcal{O}_{1,\lambda}\}$ are $(\lambda'_0, S'(\cdot))$ -indistinguishable,*

$$\mathcal{O}_{b,\lambda} = \left((\Pi_b, x_b, T_b, m_b, n_b, l_b) \stackrel{\$}{\leftarrow} \mathcal{D}_{b,\lambda} : \Pi_b^{T_b}(x_b), |\Pi_b|, |x_b|, T_b, m_b, n_b, l_b \right)$$

the ensembles of encoding $\{\mathcal{E}_{0,\lambda}\}$ and $\{\mathcal{E}_{1,\lambda}\}$ defined below is $(\lambda'_0, S''(\cdot))$ indistinguishable, where $S''(\lambda) = \frac{S'(\lambda)}{\lambda^c} - B(\lambda)^c$.

$$\begin{aligned} \mathcal{E}_{b,\lambda} = & \left((\Pi_b, x_b, T_b, m_b, n_b, l_b) \stackrel{\$}{\leftarrow} \mathcal{D}_{0,\lambda}, \right. \\ & \left. (\text{crs}, \text{pk}) \stackrel{\$}{\leftarrow} \text{Setup}(1^\lambda, 1^{m_b}, 1^{n_b}, 1^{T_b}, 1^{l_b}), \hat{\Pi}_x \stackrel{\$}{\leftarrow} \text{Enc}(\text{pk}, \Pi_b, x_b) : (\text{crs}, \text{pk}, \hat{\Pi}_x) \right) \end{aligned}$$

(We note that [63] previously defined a notion of distributional indistinguishability security for reusable garbled circuits. In the CRS model, reusable garbled circuits are equivalent to a *secret key* variant of RE (where one needs a secret key related to the CRS in order to encode). Thus our definition of distri-

butional indistinguishability of RE in the CRS model (modified to have secret key-based encoding) can be viewed as similar to their definition.)

In the CRS model, it is possible to have a compact RE for all Turing machines with simulation security. Therefore, we define compactness in the CRS model independent of security notion.

Definition 23 (Compactness and Sublinear Compactness in the CRS model). *A randomized encoding scheme $RE = (\text{Setup}, \text{Enc}, \text{Eval})$ for Turing machines in the CRS model is compact (or sublinear compact) if Setup is PPT, and Enc and Eval have the same efficiency as their counterparts in a compact (or sublinear compact) randomized encoding scheme for Turing machines in the plain model.*

Remark 2. *As mentioned in Section 4.3.5, in the plain model (λ_0, S) -simulation security implies (λ_0, S) -indistinguishability security. We remark that the same holds in the CRS model and the proof is essentially the same. We omit the details here.*

4.6.2 Succinctness and Weak Sublinear Compactness

We also consider a different weakening of compactness, called succinctness [19], where encoding time can depend linearly on the length of the output (but only polylogarithmically on the time bound T).

Definition 24 (Succinct Randomized Encoding for Turing machines [19]). *A succinct randomized encoding scheme for Turing machines in the CRS model is succinct if it has the following efficiency:*

- For every security parameters $\lambda \in \mathbb{N}$, $m, n, T, l \in \mathbb{N}$, Turing machine $\Pi \in \mathcal{M}_\lambda$ and input x , such that, $|\Pi| \leq m$, $|x| \leq n$, and $|\Pi^T(x)| \leq l$, every $(pk, crs) \leftarrow$

$\text{Setup}(1^\lambda, 1^m, 1^n, 1^T, 1^l)$ and every encoding $\hat{\Pi}_x \leftarrow \text{Enc}(1^\lambda, \Pi, x, T)$, it holds

$$\text{Time}_{\text{Setup}}(1^\lambda, 1^m, 1^n, 1^T, 1^l) = \text{poly}(\lambda, m, n, T, l)$$

$$\text{Time}_{\text{Enc}}(pk, \Pi, x) = \ell \cdot \text{poly}(\lambda, |\Pi|, |x|, \log T)$$

$$\text{Time}_{\text{Eval}}(\hat{\Pi}, \hat{x}, \text{crs}) = \text{poly}(\lambda, m, n, T)$$

We finally consider another notion of RE that is weaker than sublinear-compactness, where we allow the encoding time to be polynomially dependent on the time bound T , but still require the encoding size be sub-linear in T . We call such RE schemes *weakly sublinear compact*.

Definition 25 (Weakly Sublinear Compact Randomized Encoding scheme). *We say a randomized encoding scheme $\text{RE} = (\text{Setup}, \text{Enc}, \text{Eval})$ in the CRS model for a class of Turing machines $\{\mathcal{M}_\lambda\}$ is weakly sublinear compact if the efficiency requirement on Enc in Definition 24 is changed to: For some constant $\varepsilon \in (0, 1)$,*

$$\text{Time}_{\text{Enc}}(pk, \Pi, x) = \text{poly}(\lambda, |\Pi|, |x|, T)$$

$$\text{outlen}_{\text{Enc}}(pk, \Pi, x) = T^{1-\varepsilon} \cdot \text{poly}(\lambda, |\Pi|, |x|)$$

Next, we observe that RE schemes satisfying the notions defined above (*i.e.* succinctness and weak sublinear compactness) can be composed to get a RE scheme satisfying sub-linear compactness. In particular, by composing a succinct RE scheme with a weakly compact RE scheme, one can obtain a sub-linearly compact RE scheme.

Theorem 11. *Assume the existence of pseudorandom generators. If there is a succinct RE scheme and a weakly sublinear compact RE scheme for Turing machines, then there is a sub-linearly compact randomized encoding scheme for Turing machines.*

Proof. Let $\text{RE}_1 = (\text{Setup}_1, \text{Enc}_1, \text{Eval}_1)$ be a succinct RE scheme, and $\text{RE}_2 = (\text{Setup}_2, \text{Enc}_2, \text{Eval}_2)$ be a weakly sublinear compact RE scheme. Our sub-linearly compact scheme $\text{RE} = (\text{Setup}, \text{Enc}, \text{Eval})$ is as follows: the encoding of a machine Π is a succinct encoding of a machine G that itself computes a weakly compact encoding of Π . Weak compactness ensures the output length of G is sub-linear in ℓ (i.e. the output length of Π) and hence the time to encode G is also sub-linear in ℓ . Details follow.

$$\begin{aligned} \text{Setup}(1^\lambda, 1^m, 1^n, 1^T, 1^l) &: \text{Output } (pk_1, \text{crs}_1, pk_2, \text{crs}_2) \text{ where} \\ & \quad (pk_1, \text{crs}_1) \leftarrow \text{Setup}_1(1^\lambda, 1^{m_G}, 1^n, 1^{T_G}, 1^{l_G}) \\ & \quad (pk_2, \text{crs}_2) \leftarrow \text{Setup}_2(1^\lambda, 1^m, 1^n, 1^T, 1^l) \\ \text{Enc}(pk_1, pk_2, \Pi, x) &: s \xleftarrow{\$} \{0, 1\}^\lambda \\ & \quad \hat{G}_x \xleftarrow{\$} \text{Enc}_1(pk_1, G, x) \\ & \quad \text{where } G[\lambda, \Pi, s](x) = \text{Enc}_2(pk_2, \Pi, x, \text{PRG}(s)) \\ \text{Eval}(\text{crs}_1, \text{crs}_2, \hat{G}_x) &: \hat{\Pi}_x = \text{Eval}_1(\text{crs}_1, \hat{G}_x), y = \text{Eval}_2(\text{crs}_2, \hat{\Pi}_x) \end{aligned}$$

where T_G, m_G, l_G are upper bounds on the run-time, description size and output length of G , which can be efficiently calculated using similar bounds on Enc_2 and PRG .

It was shown in [84, 6] that randomized encoding schemes with simulation security are composable, i.e. the composed scheme defined above is also simulation secure. We note that their proof essentially goes through in the CRS model too. For concreteness, the simulator is as follows, where Sim_1 and Sim_2 are the

simulators for RE_1 and RE_2 respectively:

$$\begin{aligned} \text{Sim}(1^\lambda, \Pi^T(x), 1^{|\Pi|}, 1^{|x|}, 1^m, 1^n, 1^T, 1^l) : \text{Output } (pk_1, \text{crs}_1, pk_2, \text{crs}_2, \hat{G}_x) \text{ where} \\ (pk_2, \text{crs}_2, \hat{\Pi}_x) \leftarrow \text{Sim}_2(1^\lambda, \Pi^T(x), 1^{|\Pi|}, 1^{|x|}, 1^m, 1^n, 1^T, 1^l) \\ (pk_1, \text{crs}_1, \hat{G}_x) \leftarrow \text{Sim}_1(1^\lambda, \hat{\Pi}_x, 1^{|\Pi|}, 1^{|x|}, 1^{m_G}, 1^n, 1^{T_G}, 1^{l_G}) \end{aligned}$$

The output of Sim is indistinguishable from the real encodings, by a hybrid argument. We sketch the order of the hybrids as follows: starting with the real encoding, we first replace $(pk_1, \text{crs}_1, \hat{G}_x)$ with a simulation using Sim_1 on desired output $\hat{\Pi}_x$, where $\hat{\Pi}_x$ is honestly generated using Enc_2 and randomness from $\text{PRG}(s)$. Next, we replace $\hat{\Pi}_x$ so that it is generated using Enc_2 , but using fresh randomness instead of that from $\text{PRG}(s)$. Finally, we replace $\hat{\Pi}_x$ (and also (pk_2, crs_2)) with a simulation using Sim_2 on desired output $\Pi^T(x)$. By the security of Sim_1 , PRG and Sim_2 , these hybrids are indistinguishable.

It remains to analyze the efficiency. By construction, the time complexity of Enc is:

$$\text{Time}_{\text{Enc}}(pk_1, pk_2, \Pi, x) = \text{Time}_{\text{Enc}_1}(pk_1, G, x) = \ell_G \cdot \text{poly}(\lambda, m_G, n, \log T_G)$$

Note that by the definition of G ,

$$\begin{aligned} \ell_G &= \text{outlen}_{\text{Enc}_2}(pk_2, \Pi, x) = T^{1-\epsilon} \cdot \text{poly}(\lambda, |\Pi|, |x|) \\ T_G &= \text{Time}_{\text{Enc}_2}(pk_2, \Pi, x) = \text{poly}(\lambda, |\Pi|, |x|, T) \\ m_G &= |G[\lambda, \Pi]| = \text{poly}(\lambda, |\Pi|) \end{aligned}$$

Hence, we obtain that

$$\begin{aligned} \text{Time}_{\text{Enc}}(pk_1, pk_2, \Pi, x) &= \ell_G \cdot \text{poly}(\lambda, m_G, n, \log T_G) \\ &= T^{1-\epsilon} \cdot \text{poly}(\lambda, |\Pi|, |x|) \cdot \text{poly}(\lambda, m_G, n, \log T_G). \end{aligned}$$

Bringing all this together, using that m_G is polynomial is $\text{poly}(\lambda, |\Pi|)$, and choosing appropriate ϵ' with $0 < \epsilon' < \epsilon$, we have that:

$$\text{Time}_{\text{Enc}}(pk_1, pk_2, \Pi, x) = \text{poly}(\lambda, |\Pi|, |x|) \cdot T^{1-\epsilon'}.$$

This concludes the theorem. □

4.6.3 Randomized encodings with CRS from Compact Functional Encryption

In this section we construct RE schemes in the CRS model from Compact Functional encryption schemes and pseudorandom generators.

Let $(\text{FE.Setup}, \text{FE.Enc}, \text{FE.Dec})$ be a public key, compact functional encryption scheme for \mathcal{P}/poly , and let PRG be a pseudorandom generator. We define a randomized encoding scheme in the CRS model $(\text{Setup}, \text{Enc}, \text{Eval})$ as follows.

The setup algorithm $\text{Setup}(1^\lambda, 1^m, 1^n, 1^T, 1^l)$:

- Setup first generates keys for the functional encryption scheme $(mpk, msk) \leftarrow \text{FE.Setup}(1^\lambda)$ and samples a uniformly random string $s \leftarrow \{0, 1\}^\lambda$.
- Next, it generates the string $c \leftarrow 0^l \oplus \text{PRG}(s, l)$. That is, it encrypts 0^l using a one-time pad with the key coming from $\text{PRG}(s, l)$
- Let U be the universal circuit that on input (Π, x) where $|\Pi| \leq m$ and $|x| \leq n$ runs machine Π on x for at most T steps and outputs the first l bits of the tape as output. We define a circuit $C_{U,c}$ that has the string c and circuit U hardcoded in it, as follows.

1. $C_{U,c}$ takes as input (Π, x, s', b) where (Π, x) satisfies the size constraints as described above, $s' \in \{0, 1\}^\lambda$ and $b \in \{0, 1\}$.
 2. If $b = 0$ then $C_{U,c}$ outputs $U(\Pi, x)$.
 3. Otherwise $C_{U,c}$ outputs $c \oplus \text{PRG}(s')$.
- Setup runs $sk_C \leftarrow \text{FE.KeyGen}(msk, C_{U,c})$ and outputs sk_C as the common reference string crs and mpk as the public encoding key pk

The encoding algorithm $\text{Enc}(pk, \Pi, x)$: Enc parses pk as the functional public key mpk and runs $ct \leftarrow \text{FE.Enc}(mpk, (\Pi, x, 0^\lambda, 0))$. Enc outputs the functional ciphertext ct as the encoding $\hat{\Pi}_x$.

The evaluation algorithm $\text{Eval}(\hat{\Pi}_x, \text{crs})$: Eval parses $\hat{\Pi}_x$ as a functional ciphertext ct and crs as the functional secret key $sk_{C_{U,c}}$. Eval runs $y \leftarrow \text{FE.Dec}(sk_{C_{U,c}}, ct)$ and outputs y .

The correctness of the above encoding scheme follows directly from that of the underlying functional encryption scheme. When a randomized encoding of (Π, x) is evaluated, it outputs the result of running the universal circuit U on (Π, x) that is $\Pi^T(x)$. Also the efficiency properties of the above scheme follow directly from the compactness properties of the functional encryption scheme. For example, if the functional encryption scheme we start from has sub-linear compactness (the ciphertext size is sub-linear in the circuit size of the function for which the functional secret keys are generated) then we get an encoding scheme with sub-linear compactness.

We have the following theorem.

Theorem 12. *Let $(\text{FE.Setup}, \text{FE.Enc}, \text{FE.Dec})$ be a public key functional encryption scheme for \mathcal{P}/poly with $(\lambda_0, S(\cdot))$ selective security, and let PRG be a pseudorandom*

generator with $(\lambda_0, S(\cdot))$ security. The randomized encoding scheme defined above is $(\lambda_0, \frac{S(\cdot)}{4})$ -simulation secure.

Corollary 1. *If there exists a public key, compact (resp. succinct, weakly sublinear compact) functional encryption for P/poly scheme with selective security, and a secure PRG, then there exists a compact (resp. succinct⁸, weakly sublinear compact) randomized encoding scheme for Turing machines in the CRS model that is simulation secure.*

Proof. Let $(\text{Setup}, \text{Enc}, \text{Eval})$ be the randomized encoding scheme as defined above. We need to show there exists a PPT algorithm Sim that, when given the output of a machine on some input (together with the bound parameters of the machine and input), simulates an encoding that is indistinguishable from the real encoding of the machine and input. We define Sim as follows. **The**

simulator $\text{Sim}(1^\lambda, 1^m, 1^n, 1^T, y, 1^{|\Pi|}, 1^{|\chi|})$: Here we denote

- Sim first generates functional keys $(mpk, msk) \leftarrow \text{FE.Setup}(1^\lambda)$ and a random string $s \leftarrow \{0, 1\}^\lambda$.
- Next, Sim generates the string $c \leftarrow y \oplus \text{PRG}(s, |y|)$. That is, it encrypts the output y under a one-time pad with the key coming from $\text{PRG}(s, |y|)$.
- Sim runs $sk_C \leftarrow \text{FE.KeyGen}(msk, C_{U,c})$ where $C_{U,c}$ is same as in the construction of the encoding scheme, except with c being the string generated in the above step.
- Sim generates $ct \leftarrow \text{FE.Enc}(mpk, (0^{|\Pi|}, 0^{|\chi|}, s, 1))$.
- Sim outputs sk_C as the simulated crs, mpk as the public encoding key pk and ct as simulated machine encoding $\hat{\Pi}_x$.

⁸We note that for succinct RE, we first apply the transformation from succinct FE to get succinct RE with 1-bit output, and to encode Turing Machines with multi-bit outputs, we generate one such RE for each output bit

Now we formally prove that the above simulator is secure. Consider any ensemble $\{\Pi_\lambda, x_\lambda, m_\lambda, n_\lambda, l_\lambda, T_\lambda\}$ where $\Pi_\lambda \in \mathcal{M}_\lambda$ and $|\Pi_\lambda|, |x_\lambda|, m_\lambda, n_\lambda, l_\lambda, T_\lambda \leq B(\lambda)$ for some polynomial B . Subsequently, we suppress the security parameter in the subscript.

We need to show that for every $\lambda > \lambda_0$,

$$D_0 = \left(\begin{array}{l} (pk, crs) \xleftarrow{\$} \text{Setup}(1^\lambda, 1^{m(\lambda)}, 1^{n(\lambda)}, 1^{T(\lambda)}) \\ (\hat{\Pi}_x) \xleftarrow{\$} \text{Enc}(pk, \Pi, x) \end{array} : crs, pk, \hat{\Pi}_x \right)$$

$$D_1 = \left((pk, crs, \hat{\Pi}_x) \xleftarrow{\$} \text{Sim}(1^\lambda, 1^{m(\lambda)}, 1^{n(\lambda)}, 1^{T(\lambda)}, \Pi^T(x), 1^{|\Pi|}, 1^{|x|}) : crs, pk, \hat{\Pi}_x \right)$$

are $\frac{S(\lambda)}{4} - B(\lambda)^d$ indistinguishable, where $\lambda_0, S(\cdot)$ is the security of functional encryption scheme, and d is some constant. We show this by a hybrid argument as follows.

- Let H_0 be the distribution of the real encoding D_0 . Rewriting H_0 in terms of the underlying primitives we have

$$H_0 = \left(\begin{array}{l} (mpk, msk) \leftarrow \text{FE.Setup}(1^\lambda) \\ s \leftarrow \{0, 1\}^\lambda \\ c \leftarrow 0^l \oplus \text{PRG}(s, l) \\ sk_{C_{U,c}} \leftarrow \text{FE.KeyGen}(msk, C_{U,c}) \\ ct \leftarrow \text{FE.Enc}(mpk, (\Pi, x, 0^\lambda, 0)) \end{array} : mpk, sk_{C_{U,c}}, ct \right)$$

- Let H_1 be a hybrid distribution exactly as above, except that c is generated as $c \leftarrow 0^l \oplus R$ where R is uniformly random from $\{0, 1\}^l$. We claim H_0 and H_1 are $S(\lambda) - B(\lambda)^{d'}$ indistinguishable, where d' is some constant. This follows from the security of the pseudorandom generator. Any adversary A that distinguishes H_0 and H_1 can be turned into an adversary A' that breaks the security of the pseudorandom generator with the same advantage. A' has

Π, x, T hard-coded and needs to run the setup and generation algorithms of the functional encryption schemes. Therefore the size of A' is

$$\text{size}(A) + \text{poly}(B(\lambda)) = (S(\lambda) - B(\lambda)^{d'}) + B(\lambda)^{d'} = S(\lambda)$$

Hence A' breaks the $(\lambda_0, S(\cdot))$ security of the pseudorandom generator and we have a contradiction.

- Let H_2 be a hybrid distribution just as above except that c is generated from the output y as $c \leftarrow y \oplus R$ where, as before, R is uniformly random from $\{0, 1\}^l$. Since R is uniformly random, the distributions H_2 and H_1 are identical.
- Let H_3 be a hybrid distribution just as above except that the one-time pad key is generated using the pseudorandom generator. That is, $c \leftarrow y \oplus \text{PRG}(s, l)$. As before, the distributions H_3 and H_2 are $S(\lambda) - B(\lambda)^{d'}$ indistinguishable, where d' is some constant.
- Let H_4 be a hybrid distribution just as above except that ct is generated as

$$ct \leftarrow \text{FE.Enc}(mpk, (0^{m(\lambda)}, 0^{n(\lambda)}, s, 1))$$

This is exactly the distribution as generated by the simulator Sim . We claim H_3 and H_4 are $S(\lambda) - B(\lambda)^{d''}$ indistinguishable where d'' is some constant. This follows from the security of the functional encryption scheme. Any adversary A that distinguishes H_3 and H_4 can be turned into an adversary A' that breaks the security of the functional encryption scheme with the same advantage. A' selects the challenge messages as $(0^{m(\lambda)}, 0^{n(\lambda)}, s, 1)$ and $(\Pi, x, 0^l, 0)$ and secret key query $C_{U,c}$ which has the same output y on both messages. A' needs to additionally run the pseudorandom generator. Hence the size of A' is $\text{size}(A) + B(\lambda)^{d''}$ for some constant d'' , which as before

results in size $S(\lambda)$. Hence A' breaks the $(\lambda_0, S(\cdot))$ security of the functional encryption scheme and we have a contradiction.

Each of the hybrid distribution pairs above are $S(\lambda) - B(\lambda)^d$ indistinguishable. By a simple hybrid argument we have that the distributions $D_0 = H_0$ and $D_1 = H_4$ are $\frac{S(\lambda) - B(\lambda)^d}{4} \geq \frac{S(\lambda)}{4} - B(\lambda)^d$ indistinguishable, for some constant d , hence completing the proof.

□

The above theorem and corollary also work in the regime of sub-exponential security. That is, starting with a functional encryption scheme and pseudorandom generator that are sub-exponentially secure we obtain a RE scheme with sub-exponential security.

The following corollary is obtained by combining Corollary 1 with Theorem 6 and Theorem 7. While we use this corollary in our results, we believe it is of independent interest too. Succinct RE schemes for Turing machines were shown by [19] to have a variety of applications. However the only known construction of it ([90]) relies on iO for circuits. We observe that in the CRS model, succinct RE schemes can be based simply on LWE.

Corollary 2. *Assuming LWE (resp. with sub-exponential hardness), there exists a succinct RE scheme for Turing machines in the CRS model with (resp. sub-exponential) simulation security.*

Finally, the following corollary shows that, assuming LWE, weakly sublinear compact FE is sufficient to construct sublinearly-compact RE in the CRS model.

This corollary follows by combining Corollary 1, which shows that weakly sub-linear compact FE implies weakly sublinear compact RE in the CRS model, Corollary 2, which constructs succinct RE in the CRS model from LWE, and finally Theorem 11, which shows that weakly sublinear compact RE and succinct RE can be combined to produce sublinearly-compact RE in the CRS model.

Corollary 3. *Assuming LWE (resp. with sub-exponential hardness), if there exists a weakly sublinear compact FE scheme for $P/poly$ (resp. with sub-exponential security), then there exists a sublinearly-compact RE scheme for Turing machines in the CRS model with (resp. sub-exponential) simulation security.*

4.6.4 IO for Circuits from RE in the CRS model

In this section we show that compact RE schemes for Turing machines in the CRS model implies iO for circuits; combining with the result of [90] that iO for circuits implies iO for (bounded-input) Turing machines, we obtain the following theorem:

Theorem 13. *Assume the existence of sub-exponentially secure one-way functions. If there exists a sublinearly compact randomized encoding scheme in the CRS model with sub-exponential simulation security, then there exists an bounded-input indistinguishability obfuscator for Turing machines.*

We note that the theorem also holds w.r.t. sublinearly compact randomized encoding scheme in the CRS model, satisfying, weaker, distributional indistinguishability security, *with auxiliary inputs* (i.e., Definition 22 w.r.t. distributions $\{D_{b,\lambda}\}$ that additionally samples an auxiliary input z_b , and the security requirement is that if the output distributions together with the auxiliary inputs are in-

distinguishable, then the encodings together with the auxiliary inputs are also indistinguishable, with appropriate security loss). Since the distributional indistinguishability security is implied by simulation security, and in the CRS model, we can construct sublinearly compact RE with simulation security from sublinearly compact FE schemes, for simplicity, we directly state and prove the theorem w.r.t. simulation security.

The construction and proof is very similar to that of unbounded-input $i\mathcal{O}$ from compact RE schemes in the plain model presented in Section 4.4.

Proof sketch for Theorem 13. We first briefly recall the main ideas behind the unbounded-input $i\mathcal{O}$ construction. The unbounded-input $i\mathcal{O}$ generates an encoding of a recursively defined Turing machine $\Pi_{\epsilon,R}$ (where ϵ is the empty string and R is uniformly random). $\Pi_{s,R}$ generates encodings of Π_{s0,R_0} , Π_{s1,R_1} and of the machine to be obfuscated with input s , where R_0 and R_1 are pseudorandom strings derived from R . For every such machine $\Pi_{s,R}$, we refer to $|s|$ as the *level* of the machine. Evaluating the obfuscation on an input of length n involves evaluating an encoding of a machine at every level $i \in \{0, \dots, n\} = [n]$.

Construction: Our circuit obfuscator $i\mathcal{O}$ gets as input the security parameter 1^λ and the circuit to obfuscate C . Let n be the input length of C . To use RE schemes in the CRS model, $i\mathcal{O}$ first generates a RE setup (pk_i, crs_i) for every level $i \in [n]$. The obfuscation consists of $\vec{crs} = \{crs_i\}_{i=0}^n$, and an encoding of the machine $\Pi_{\vec{pk}_1, C, \epsilon, R}$ which has public keys $\vec{pk}_1 = \{pk_i\}_{i=1}^n$ hardcoded (more generally \vec{pk}_i denotes $\{pk_j\}_{j=i}^n$). For every level $i < n$ and $s \in \{0, 1\}^i$, the machine $\Pi_{\vec{pk}_{i+1}, C, s, R}$ uses pk_{i+1} to generate encodings of $\Pi_{\vec{pk}_{i+2}, C, s0, R_0}$ and $\Pi_{\vec{pk}_{i+2}, C, s1, R_1}$. When $i = n$, the machine $\Pi_{\vec{pk}_{i+1}, C, s, R}$ simply outputs $C(s)$. To evaluate the obfuscation on an input x , one evaluates an encoding of the machine $\Pi_{\vec{pk}_{i+1}, C, x[1\dots i], R}$ at every level $i \in [n]$,

using $\text{crs}_i \in \vec{\text{crs}}$.

Note that, just as in the unbounded-input $i\mathcal{O}$ construction (Lemma 3), the compactness of the RE ensures that the size of the encodings at each level is some fixed polynomial in the security parameter and the circuit size. The obfuscation additionally contains a crs_i for every level $i \in [n]$ where the length of crs_i is polynomial in the running time of machines at that level (*i.e.* the time taken to encode machines at the next level), which by the same argument (Lemma 3) is some fixed polynomial in the security parameter and the circuit size. All in all, the size of the obfuscated circuit is polynomial in the security parameter and the circuit size.

Security: We need to show that, for any pair of functionally equivalent circuits C_0 and C_1 , the joint distribution $(\vec{\text{crs}}, \tilde{\Pi}_{pk_1, \epsilon, C_0, R})$ is indistinguishable from $(\vec{\text{crs}}, \tilde{\Pi}_{pk_1, \epsilon, C_1, R})$. Just as in the proof of Theorem 10, we prove a stronger statement by induction. We claim that for every level $i \in [n]$, and every $s \in \{0, 1\}^i$, the joint distribution $(\tilde{\Pi}_{pk_{i+1}, s, C_0, R}, \vec{\text{crs}}_i, \vec{pk}_i)$ is indistinguishable from $(\tilde{\Pi}_{pk_{i+1}, s, C_1, R}, \vec{\text{crs}}_i, \vec{pk}_i)$.

When $i = n$ (the base case), the above distributions are indistinguishable since the output of $\Pi_{pk_{i+1}, s, C_0, R}$ (in this case $C_0(s)$) is identical to the output of $\Pi_{pk_{i+1}, s, C_1, R}$. Hence by the indistinguishability security of the RE scheme (which is implied by simulation security, see Remark 2), $(\tilde{\Pi}_{pk_{n+1}, s, C_0, R}, pk_n, \text{crs}_n)$ is indistinguishable from $(\tilde{\Pi}_{pk_{n+1}, s, C_1, R}, pk_n, \text{crs}_n)$.

For the inductive step, we show the distributions are indistinguishable at any level $i < n$ assuming they are indistinguishable at level $i + 1$. We do this by a hybrid argument as follows. Let H_0 be the joint distribution of the encoding at level i with C_0 hardcoded, with $\vec{\text{crs}}_i$ and \vec{pk}_i . Writing this a bit differently, we

have,

$$H_0 = ((\tilde{\Pi}_{pk_{i+1},s,C_0,R}, \text{crs}_i, pk_i), \text{crs}_{i+1}, \vec{pk}_{i+1})$$

Next, we define hybrid distribution H_1 as follows.

$$H_1 = (\text{Sim}(\text{out}(\tilde{\Pi}_{pk_{i+1},s,C_0,R})), \text{crs}_{i+1}, \vec{pk}_{i+1})$$

where Sim is the simulator for the RE scheme (for the sake of brevity in this proof sketch, we omit the other inputs to the simulator). By simulation security, H_0 and H_1 are indistinguishable.

Next, we define hybrid distribution H_2 by changing the underlying circuit to C_1 .

$$H_2 = (\text{Sim}(\text{out}(\tilde{\Pi}_{pk_{i+1},s,C_1,R})), \text{crs}_{i+1}, \vec{pk}_{i+1})$$

To show H_2 and H_1 are indistinguishable, we show that the following distributions are indistinguishable.

$$(\text{out}(\Pi_{pk_{i+1},s,C_0,R}), \text{crs}_{i+1}, \vec{pk}_{i+1}) \approx (\text{out}(\Pi_{pk_{i+1},s,C_1,R}), \text{crs}_{i+1}, \vec{pk}_{i+1})$$

This follows from the induction hypothesis as follows. Recall that the output of $\Pi_{pk_{i+1},s,C_0,R}$ is a pair of level $i + 1$ encodings $(\tilde{\Pi}_{pk_{i+2},s0,C_0,R_0}, \tilde{\Pi}_{pk_{i+2},s1,C_0,R_1})$. By the induction hypothesis, each of these encodings is indistinguishable from one with C_1 hardcoded, in the presence of $(\text{crs}_{i+1}, \vec{pk}_{i+1})$. By a simple hybrid argument, the above indistinguishability holds, and hence H_2 is indistinguishable from H_1 .

Finally, we define hybrid distribution H_3 which contains the encoding at level i with C_1 hardcoded.

$$H_3 = ((\tilde{\Pi}_{pk_{i+1},s,C_1,R}, \text{crs}_i, pk_i), \text{crs}_{i+1}, \vec{pk}_{i+1})$$

By the simulation security of the RE scheme H_3 is indistinguishable from H_2 . Hence, by a hybrid argument H_0 is indistinguishable from H_3 hence completing the inductive step. \square

4.6.5 Summary of Results using RE in the CRS model

We observe that by combining Theorem 13 with Corollary 1, we reprove the results of [3, 22]

Theorem 14. *Assuming the existence of compact functional encryption with subexponential security, there exists a bounded-input indistinguishability obfuscator for Turing Machines.*

Further, we get the following new result, as a consequence of Corollary 3 and Theorem 13:

Theorem 15. *Assuming the existence of weakly sublinear compact functional encryption with subexponential security and LWE with subexponential security, there exists a bounded-input indistinguishability obfuscator for Turing Machines.*

4.7 Impossibility of Compact RE

In this section, we show several impossibility results related to sublinear (and hence compact) RE with different security:

Theorem 16. *The following impossibility results hold in the plain model:*

1. *Sublinear randomized encoding schemes with (polynomial) simulation security do not exist, assuming one-way functions.*
2. *Sublinear randomized encoding schemes with sub-exponential indistinguishability security do not exist, assuming sub-exponentially secure one-way functions.*
3. *Sublinear randomized encoding schemes with (polynomial) indistinguishability security do not exist, assuming bounded-input iO for Turing machines and one-way functions.*

Next we proceed to prove Theorem 16.

Impossibility 1. The impossibility of sublinear RE with simulation security follows from standard techniques that leverages the sublinear-size of the encoding to derive a contradiction to the incompressibility of pseudo-random strings; below, we provide a proof sketch.

Proof Sketch. We argue that assuming one-way functions, compact RE with simulation security does not exist in the plain model. Suppose not and there is a compact RE that admits a simulator Sim which on input the output $y = \Pi(x)$ can simulate an encoding $(\tilde{\Pi}, \tilde{x})$ that is indistinguishable from an honestly generated encoding $(\hat{\Pi}, \hat{x})$. By the indistinguishability, it follows that evaluating $(\tilde{\Pi}, \tilde{x})$ yields the output y . Now, consider the specific computation of evaluating a PRG G on a short random seed s , that is, $\Pi = G$ and $x = s$. The simulator Sim on the pseudo-random output $y = G(s)$, produces (\tilde{G}, \tilde{s}) , that can be evaluated to generate y . By the pseudo-randomness of PRG, it follows that Sim on input a truly random string y' , can also output a tuple (\tilde{G}', \tilde{s}') that evaluates to y' , and

the length of the tuple is sublinear in the length of y' . However, this contradicts the incompressibility of random strings. \square

Impossibility 2. The impossibility of sub-exponentially (ind-)secure sublinear RE is implied by impossibility 3, and the fact that bounded-input iO for Turing machines can be constructed from sub-exponentially (ind-)secure sublinear RE. More precisely, by our construction in Section 4.5, sub-exp secure sublinear-RE (and one-way functions) imply sub-exp secure iO for circuits; combined with the result of [90] that sub-exp secure iO for circuits (and one-way functions) imply (polynomially secure) bounded-input iO for Turing machines, we have that assuming sub-exp secure one-way functions,

$$\text{Sub-exp (ind-)secure Sublinear RE} \implies \text{Bounded-input } iO \text{ for TMs}$$

On the other hand, impossibility 3 states that

$$\text{Bounded-input } iO \text{ for TMs} \implies \text{NO (ind-)secure Sublinear RE}$$

Therefore, if impossibility 3 is true, sub-exponentially secure sublinear RE is impossible assuming sub-exp secure one-way functions.

Impossibility 3. We show that (poly-secure) sublinear RE does not exist assuming the following two primitives.

- A bounded-input iO for Turing machines iO , which on input $(1^\lambda, 1^n, R, T)$ runs in time

$$\text{Time}_{iO}(1^\lambda, 1^n, R, T) \leq \text{poly}(\lambda, n, |R|, \log T).$$

- A pseudo-random generator PRG that on input a seed s of length λ and a length k outputs a string of length k in time $\text{poly}(\lambda, k)$. This is implied by the existence of one-way functions.

The efficiency of iO and PRG means that there is a constant d , such that, the following holds:

- For every $\lambda \in N$, $\ell = \ell(\lambda)$, program R with size $|R| \leq \lambda + \ell$, input length $n = \ell/2$, and $T = 2^\lambda$, the run-time of iO is

$$\text{Time}_{iO}(1^\lambda, 1^{\ell/2}, R, 2^\lambda) \leq (\lambda\ell)^d, \quad (4.1)$$

- and for every string $s \in \{0, 1\}^\lambda$, and $k \in N$, the run-time of the PRG is

$$\text{Time}_{\text{PRG}}(s, k) \leq (\lambda k)^d. \quad (4.2)$$

Assuming such iO and PRG, we first show that there does not exist sublinear RE that are sufficiently compact in the following sense:

Claim 2. *Let d be a constant defined as above w.r.t. iO and PRG. There is no sublinear RE with time complexity satisfying the following:*

$$\text{Time}_{\text{Enc}}(1^\lambda, \Pi, x, T) \leq \text{poly}(\lambda, |\Pi|, |x|)T^\alpha, \quad \text{for } \alpha \leq 1/2d. \quad (4.3)$$

Before proving the claim, we argue that the above claim suffices for ruling out the existence of any sublinear RE schemes. By Claim 2 in Section 4.3.4, given any sublinear RE with complexity scaling with T^β for an arbitrary constant $\beta > 0$, one can reduce the time complexity to scaling with T^α for an arbitrary smaller constant α , by recursively composing it for some constant times. Combining this fact with the above claim, we conclude that assuming bounded-input iO for TMs and one-way function, sublinear RE do not exist.

Proof of Claim 2. Assume for contradiction that there is a sublinear RE scheme $\text{RE} = (\text{Enc}, \text{Eval})$ whose time complexity depends on T^α with a sufficiently small $\alpha \leq 1/2d$ (and some multiplicative polynomial factor $\text{poly}(\lambda, |\Pi|, |x|)$). Below, we derive a contradiction by constructing two ensembles of distributions $\{D_{0,\lambda}\}, \{D_{1,\lambda}\}$ that both sample triplets of form (Π, x, T) ; we show that (1) the outputs of the program and input sampled from these two distributions are indistinguishable, yet (2) the encoding of the program and input are distinguishable with probability close to 1. This violates the ind-security of RE, and gives a contradiction.

Let $\ell = \ell(\lambda)$ be a sufficiently large polynomial in λ , whose magnitude will become clear in the description below.

Distribution $D_{b,\lambda}$ samples triplet $(\Pi_b, 0, T)$ as follows:

- Sample two seeds $s \xleftarrow{\$} \{0, 1\}^\lambda$, and $u \xleftarrow{\$} \{0, 1\}^\lambda$.
- Turing machine $\Pi_b[\lambda, s, u]$ (with (λ, s, u) hardwired in), on input 0, proceeds in two steps
 1. Compute $\text{PRG}(s, \ell) = y$, and $\text{PRG}(u, \Gamma) = r$ where $\Gamma = (\lambda\ell)^d$.
 2. Obfuscate the program $R_b[b, y]$ described in Figure 4.1 to obtain

$$\hat{R}_b = i\mathcal{O}(1^\lambda, 1^{\ell/2}, R_b, 2^\lambda; r).$$

(The input length of R_b is bounded by $\ell/2$ and its run time is bounded by 2^λ .)

3. Output (y, \hat{R}_b) .
- Set $T = 2(\lambda\Gamma)^d$.

Note that T is an upper bound on the run-time of Π_b , as the first step of Π_b

takes at most $(\lambda\ell)^d + (\lambda\Gamma)^d$ (according to condition (4.2)), and the second step takes at most $(\lambda\ell)^d$ steps (according to condition (4.1)).

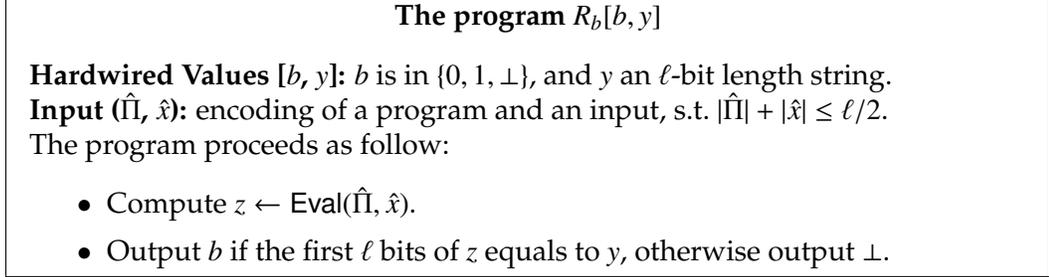


Figure 4.1: The program R_b used in the proof of impossibility of sublinear RE in the plain model.

We first show that the distributions, $out_{0,\lambda}$ and $out_{1,\lambda}$, of outputs of the program and input sampled from $D_{0,\lambda}$ and $D_{1,\lambda}$ are indistinguishable

$$\begin{aligned}
out_{b,\lambda} &= \left((\Pi_b, 0, T) \stackrel{\$}{\leftarrow} D_{b,\lambda} : \Pi_b^T(0), T, |\Pi_b|, |0| \right) \\
&= \left(s, u \stackrel{\$}{\leftarrow} \{0, 1\}^\lambda, y = \text{PRG}(s, \ell), r = \text{PRG}(u, \Gamma), \right. \\
&\quad \left. \hat{R}_b = i\mathcal{O}(1^\lambda, 1^{\ell/2}, R_b[b, y], 2^\lambda; r) : (y, \hat{R}_b), T, |\Pi_b|, |0| \right)
\end{aligned}$$

Towards this, consider the following hybrid distributions.

Distribution $H_{b,\lambda}$ samples output tuple in the same way as $out_{b,\lambda}$ does, except that the pseudo-random strings y and r are replaced with truly random strings $\tilde{y} \stackrel{\$}{\leftarrow} \{0, 1\}^\ell$ and $\tilde{r} \stackrel{\$}{\leftarrow} \{0, 1\}^\Gamma$. More precisely,

$$\begin{aligned}
H_{b,\lambda} &= \left(\tilde{y} \stackrel{\$}{\leftarrow} \{0, 1\}^\ell, \tilde{r} \stackrel{\$}{\leftarrow} \{0, 1\}^\Gamma, \right. \\
&\quad \left. \tilde{R}_b = i\mathcal{O}(1^\lambda, 1^{\ell/2}, R_b[b, \tilde{y}], 2^\lambda; \tilde{r}) : (\tilde{y}, \tilde{R}_b), T, |\Pi_b|, |0| \right)
\end{aligned}$$

By the security of PRG, the above distribution is indistinguishable from $out_{b,\lambda}$.

Distribution $G_{b,\lambda}$ samples output tuple in the same way as $H_{b,\lambda}$ does, except that, instead of obfuscating the program R_b , it obfuscate the program R_\perp , which always outputs \perp (and has the same run time as R_b). More precisely,

$$G_{b,\lambda} = \left(\tilde{y} \stackrel{\$}{\leftarrow} \{0, 1\}^\ell, \tilde{r} \stackrel{\$}{\leftarrow} \{0, 1\}^\Gamma, \right. \\ \left. \underline{\tilde{R}_\perp = iO(1^\lambda, 1^{\ell/2}, R_\perp[\perp, \tilde{y}], 2^\lambda; \tilde{r})} : (\tilde{y}, \underline{\tilde{R}_\perp}, T, |\Pi_b|, |0|) \right)$$

We claim that it follows from the security of iO that $G_{b,\lambda}$ and $H_{b,\lambda}$ are indistinguishable. Towards this, it suffices to show that with probability $1 - 2^{-\ell/2}$ over the random choice of \tilde{y} , the program $R_b[b, \tilde{y}]$ (obfuscated in $H_{b,\lambda}$) always outputs \perp , in which case $R_b[b, \tilde{y}]$ and $R_\perp[\perp, \tilde{y}]$ agree on all inputs and their obfuscation is indistinguishable. By construction, R_b outputs a value that is not \perp only when the input $(\hat{\Pi}, \hat{x})$ satisfies that the first ℓ bits of the string z evaluated from it matches \tilde{y} . However, the encoding length is bounded by $\ell/2$, there are at most $2^{\ell/2}$ possible string z ; when \tilde{y} is chosen at random, the probability that z agrees with \tilde{y} is at most $2^{-\ell/2}$. Except with this probability, R_b always outputs \perp .

On the other hand, we show that the distributions, $enc_{0,\lambda}$ and $enc_{1,\lambda}$, of the encoding of the program and input sampled from $D_{0,\lambda}$ and $D_{1,\lambda}$ are efficiently *distinguishable*.

$$enc_{b,\lambda} = \left((\Pi_b, 0, T) \stackrel{\$}{\leftarrow} D_{b,\lambda} : (\hat{\Pi}_b, \hat{0}) \stackrel{\$}{\leftarrow} \text{Enc}(1^\lambda, \Pi_b, 0, T) \right)$$

Towards this, we first show that the length of the encoding $(\hat{\Pi}_b, \hat{0})$ is bounded by $\ell/2$. This is because the run-time of Enc satisfies

$$\begin{aligned} \text{Time}_{\text{Enc}}(1^\lambda, \Pi_b, 0, T) &\leq \text{poly}(\lambda, |\Pi_b|, |0|) T^\alpha \text{ for } \alpha \leq 1/2d \\ &\leq \text{poly}(\lambda) (2(\lambda\Gamma)^d)^{1/2d^2} = \text{poly}(\lambda) \Gamma^{1/2d} \\ &= \text{poly}(\lambda) ((\lambda\ell)^d)^{1/2d} \\ &\leq \lambda^c \sqrt{\ell} \leq \ell/2 \end{aligned}$$

where the second line follows from that $|\Pi_b| = O(\lambda)$ and $T = 2(\lambda\Gamma)^d$, the third line follows from that $\Gamma = (\lambda\ell)^d$, and in the last line, c is a constant independent of d and the last inequality holds for sufficiently large $\ell = \ell(\lambda)$.

□

Acknowledgment: We are extremely grateful to Nir Bitansky and Omer Paneth for informing us of their impossibility result for compact RE assuming differing-input obfuscation, SNARKs and collision-resistant hash functions; this result was the inspiration behind our main impossibility result. We are also very grateful to them for many delightful and insightful discussions.

CHAPTER 5

INDISTINGUISHABILITY OBFUSCATION WITH EXPONENTIAL EFFICIENCY

This chapter contains joint work with Huijia Lin (UCSB), Rafael Pass (Cornell University) and Sidharth Telang (Cornell University), to appear in the Conference on Public Key Cryptography (PKC) 2016.

5.1 Introduction

The goal of *program obfuscation* is to “scramble” a computer program, hiding its implementation details (making it hard to “reverse-engineer”), while preserving the functionality (i.e, input/output behavior) of the program. In recent years, the notion of *indistinguishability obfuscation* (iO) [12, 57] has emerged as the central notion of obfuscation: Roughly speaking, this notion requires that obfuscations $iO(C_1)$, $iO(C_2)$ of any two *functionally equivalent* circuits C_1 and C_2 (i.e., whose outputs agree on all inputs) from some class C (of circuits of some bounded size) are computationally indistinguishable.

On the one hand, this notion of obfuscation is strong enough for a plethora of amazing applications (see e.g., [116, 32, 30, 55, 19, 43, 90]); on the other hand, it may plausibly exist [57, 9, 106, 64], whereas stronger notion of obfuscations have run into strong impossibility results, even in idealized models (see e.g., [12, 72, 44, 108, 94, 92])

However, despite all these amazing progress, to date, all candidate constructions of iO rely on candidate constructions of *multi-linear maps* [54, 49, 61, 51], all

of which have non-trivial attacks [47, 97], and it is not clear to what extent the security of the obfuscators that rely on them are affected.

In this paper, rather than studying new candidate construction of iO , we are interested in studying *weaker* notions of indistinguishability obfuscation that can amplified/bootstrapped into the standard notion of iO :

Identify quantitatively weaker notions of iO that can be amplified into the “standard” notion of iO .

Our hope that that doing so will simplify future constructions of iO . This approach is orthogonal to the approach of [57], which studies whether iO for “weak” classes of circuits (e.g., NC^1 circuits) can be bootstrapped to iO for all polynomial-size circuits, but is similar to e.g., the hardness amplification approach of Yao [117] of amplifying a *weak one-way function* into a (strong) one.)

One initial weakening of iO appeared (implicitly) in [19], where the authors consider iO for polynomial-size circuits with $O(\log \lambda)$ length inputs, where λ is the security parameter; we refer to this class of circuits as $P^{O(\log \lambda)}/\text{poly}$ and refer to iO for $P^{O(\log \lambda)}/\text{poly}$ as *short-input iO* . Short-input iO is more appealing than standard iO (for P/poly) in the sense that it can be efficiently checked whether an attack on a candidate scheme succeeds [100] (an attacker needs to come up with two circuits C_1, C_2 that are functionally equivalent for which it can distinguish obfuscations; checking whether two circuits are functionally equivalent may be hard in general, but becomes efficient if the circuits are restricted to inputs of length $O(\log \lambda)$ by simply enumerating all inputs). Additionally, [19] show that for *some* (but far from all) applications of iO , this weaker notion actually suffices. But it is not known whether this weaker notion of iO implies “standard iO ”; we

shall return to this question shortly.

Inefficient iO : We here consider a further weakening of short-input iO . Recall that indistinguishability obfuscators with running time

$$T_0(|C|, \lambda, n) = \text{poly}(|C|, \lambda) \cdot 2^n,$$

and size

$$\text{Size}_0(|C|, \lambda, n) = \text{poly}(|C|, \lambda) \cdot 2^n,$$

where C is the circuit to be obfuscated, λ is the security parameter, and n is the input length of C , exists *unconditionally*—simply output the function table of C (i.e., the output of C on all possible inputs). Such inefficient iO , however, are not useful for applications.

We here consider iO with just “slightly non-trivial” running-time; namely, we allow the running time to be

$$T_0(|C|, \lambda, n) = \text{poly}(|C|, \lambda) \cdot 2^n,$$

but require the *size* of the obfuscation to be

$$\text{Size}_\epsilon(|C|, \lambda, n) = \text{poly}(|C|, \lambda) \cdot 2^{n(1-\epsilon)}$$

where $\epsilon > 0$. We refer to this notion as iO with *exponential efficiency*, or simply *exponentially-efficient iO (XiO)* (Recall that, in contrast, for “standard” iO , the running time and size of the obfuscator is required to be $\text{poly}(|C|, \lambda)$). In essence, **XiO** requires the obfuscator to be just slightly smaller than a brute-force canonicalization of the circuit.

Note that **XiO** obfuscators are only efficiently computable for circuits that take short inputs; we thus here restrict our attention to **XiO** for $P^{O(\log \lambda)}/\text{poly}$ (or “short-input” **XiO**).

Main Theorem: Perhaps surprisingly, we show that in the regime of subexponential security, under the LWE assumption, **XiO** for $P^{O(\log \lambda)}/\text{poly}$ implies (standard) *iO* for P/poly .

Theorem 17. *Assume subexponential security of the LWE assumption, and the existence of subexponentially-secure **XiO** for $P^{O(\log \lambda)}/\text{poly}$. Then there exists subexponentially-secure *iO* for P/poly .*

As a corollary of Theorem 17, we get that subexponentially-secure short-input *iO* implies subexponentially-secure “standard” *iO* (since *iO* trivially implies **XiO**).

Techniques [92], improving on results of Ananth and Jain [3] and Bitansky and Vaikuntanathan [22], show that the existence of subexponentially-secure *functional encryption with weakly sublinearly compact ciphertexts* (a.k.a. *weakly sublinear compact FE*) for P/poly implies *iO* for P/poly . Roughly speaking, a (single-key) functional encryption scheme is a public-key encryption scheme for which it is possible to release a (single) functional secret-key sk_C (for circuit C of some a-priori bounded size S) such that knowledge of sk_C enables efficiently computing $C(m)$ given any encryption of the message m , (but nothing more); sublinear compactness means that the ciphertext *size* is sublinear in the upper bound S on the circuit-size (though the encryption *time* is allowed to depend polynomially on S).¹

Our main technical contribution will be showing that **XiO** for $P^{O(\log \lambda)}/\text{poly}$ implies sublinear compact FE for P/poly , which by the above-mentioned result implies our main theorem.

¹More precisely, in a functional encryption scheme (Setup, KeyGen, Enc, Dec), Setup samples a public-key, secret-key pair (pk, msk) , KeyGen(msk, C) generates the functional secret key sk_C ; Enc(pk, m) outputs an encryption c of m , and Dec(sk_C, c) outputs $C(m)$ if c is an encryption of m .

Theorem 18. *Assume the LWE assumption (resp. subexponential security of the LWE assumption) holds, and the existence of \mathbf{XiO} for $\mathsf{P}^{\mathcal{O}(\log \lambda)}/\text{poly}$ (resp. subexponentially-secure \mathbf{XiO} for $\mathsf{P}^{\mathcal{O}(\log \lambda)}/\text{poly}$). Then there exists weakly sublinear compact FE for P/poly (resp. subexponentially-secure sublinear compact FE for NC^1).*

Note that Theorem 18 is interesting in its own right as it applies also in the regime of polynomial security.²

The proof of Theorem 18 proceeds as follows. Following a proof template from [3] (we discuss this result in more detail below), we start off with the result of Goldwasser et al [73] which shows that under the LWE assumption, there exists a functional encryption scheme for *boolean* functions (i.e., functions with 1-bit outputs) in NC^1 that has *logarithmic* compactness. Combined with [2], this can be used to construct a functional encryption scheme for *boolean* functions in P/poly that still has logarithmic compactness. We next show how to use \mathbf{XiO} for $\mathsf{P}^{\mathcal{O}(\log \lambda)}/\text{poly}$ to extend any such compact FE scheme for boolean functions to one that handles arbitrary polynomial-sized circuits (with potentially long outputs). ([3] provided a similar transformation assuming so-called *compact randomized encoding* instead of \mathbf{XiO} .)

We now turn to describe our transformation from “single-bit compact FE” to “multi-bit weakly sublinear compact FE”. As an initial approach, instead of simply encrypting a message m , encrypt the sequence $(m; 1), (m; 2), \dots (m; \ell)$, where ℓ is the maximum output length of the class of functions we want to be able to evaluate. Then, instead on simply releasing a functional secret key for a circuit C , release a secret key for the function $C'(m; i) = C_i(m)$, where $C_i(m)$ denotes the i th

²Furthermore, as we remark later on, sublinear compact FE trivially implies a variant of \mathbf{XiO} and this variant of \mathbf{XiO} is also sufficient for our theorems. As such, by our results, \mathbf{XiO} may be viewed as a new way to characterize the complexity of sublinear compact FE.

output bit of $C(m)$. This approach clearly enables evaluating circuits with multi-bit outputs; but the encryption scheme is no longer compact! The length of the ciphertext grows *linearly* with the number of output bits. To retain compactness (or at least sublinear compactness), we have the encryption algorithm release an obfuscation of a program Π that generates all the ℓ encryptions—more precisely, given an index i , it applies a PRF (with a hard-coded seed) to the index i to generate randomness r_i and then outputs an encryption of $(m; i)$. As long as obfuscation size is “just-slightly-compressing”, the functional encryption will have weak sublinear compactness; furthermore, the program we obfuscate only needs to take inputs of length $O(\log \lambda)$. Thus, it suffices to assume the obfuscator satisfies **XiO** for $P^{O(\log \lambda)}/\text{poly}$.

To prove security of the construction, we use the “one-input-at-a-time” technique from [32, 65, 107, 64, 46], and the punctured program technique of Sahai and Waters [116]; the crucial point that enables us to keep the obfuscation small is that the output of the program Π on different inputs uses independent randomness (since they are independent encryptions) and thus in the hybrid arguments it suffices to puncture the PRF on a single point.

Let us end this section by briefly comparing our transformation to that of Ananth and Jain [3]; as mentioned above [3] shows how to use “compact randomized encoding” to transform single-bit compact FE for NC^1 into multi-bit compact FE for NC^1 . As we explain in more detail in Remark 5, compact randomized encoding can be viewed as a special case of **XiO** for the class of *Turing machines* (as opposed to circuits) with short input. Turing machine obfuscation is a significantly more challenging task than circuit obfuscation. We provide a brief description of their transformation in Section 5.5 and explain why the

transformation fails when using **XiO**.

5.2 Preliminaries

Let \mathcal{N} denote the set of positive integers, and $[n]$ denote the set $\{1, 2, \dots, n\}$. We denote by PPT probabilistic polynomial time Turing machines, and by nuPPT non-uniform probabilistic polynomial time Turing machines. The term *negligible* is used for denoting functions that are (asymptotically) smaller than one over any polynomial. More precisely, a function $\nu(\cdot)$ from non-negative integers to reals is called *negligible* if for every constant $c > 0$ and all sufficiently large n , it holds that $\nu(n) < n^{-c}$. For any algorithm A and input x we denote by $\text{outlen}_A(x)$, the output length of A when run with input x .

Definition 26. We denote by $\mathsf{P}^{\mathsf{O}(\log \lambda)}/\text{poly}$ the class of circuits $\{C_\lambda\}$ where C_λ are $\text{poly}(\lambda)$ -size circuits that have input length $c \log \lambda$ for some constant c .

5.2.1 Puncturable PRF

Definition 27 (Puncturable PRF). A puncturable pseudo-random function F is given by a triple of efficient algorithms $(F.\text{Key}, F.\text{Punc}, F.\text{Eval})$, and a pair of computable functions $n(\cdot)$ and $m(\cdot)$, satisfying the following conditions:

- **Functionality preserved under puncturing:** For every polynomial size set $S \subseteq \{0, 1\}^{n(\lambda)}$ and for every $x \in \{0, 1\}^{n(\lambda)} \setminus S$, we have that:

$$\Pr[F.\text{Eval}(K, x) = F.\text{Eval}(K_S, x) : K \leftarrow F.\text{Key}(1^\lambda), K_S = F.\text{Punc}(K, S)] = 1$$

- **Pseudorandom at punctured points:** For every polynomial size set $S \subseteq \{0, 1\}^{n(\lambda)}$ we have that for every nuPPT adversary A we have that:

$$|\Pr[A(K_S, \text{F.Eval}(K, S)) = 1] - \Pr[A(K_S, U_{m(\lambda)|S}) = 1]| = \text{negl}(\lambda)$$

where $K \leftarrow \text{F.Key}(1^\lambda)$ and $K_S = \text{F.Punc}(K, S)$ and $\text{F.Eval}(K, S)$ denotes the concatenation of $\text{F.Eval}(K, x_1), \dots, \text{F.Eval}(K, x_k)$ where $S = \{x_1, \dots, x_k\}$ is the enumeration of the elements of S in lexicographic order, U_ℓ denotes the uniform distribution over ℓ bits.

5.2.2 Functional Encryption

We note that in this work, we only need the security of the functional encryption scheme to hold with respect to statically chosen challenge messages and functions. We further consider FE schemes that only produce a single functional secret key for each public key.

Definition 28 (Functional Encryption). A public key functional encryption scheme for a class of circuits $\{C_\lambda\}$ is a tuple of PPT algorithms $(\text{FE.Setup}, \text{FE.KeyGen}, \text{FE.Enc}, \text{FE.Dec})$ that behave as follows:

- $(msk, pk) \leftarrow \text{FE.Setup}(1^\lambda)$: FE.Setup takes as input the security parameter λ and outputs the master secret key msk and public key pk .
- $sk_C \leftarrow \text{FE.KeyGen}(msk, C)$: FE.KeyGen takes as input the master secret key and a circuit $C \in C_\lambda$ and outputs the functional secret key sk_C .
- $c \leftarrow \text{FE.Enc}(pk, m)$: FE.Enc takes as input the public key and message $m \in \{0, 1\}^*$ and outputs the ciphertext c .

- $y \leftarrow \text{FE.Dec}(sk_C, c)$: FE.Dec takes as input the functional secret key and ciphertext and outputs $y \in \{0, 1\}^*$.

We require the following conditions hold:

- **Correctness:** For every $\lambda \in \mathbb{N}$, $C \in C_\lambda$ with input length n and message $m \in \{0, 1\}^n$, we have that

$$\Pr \left[\begin{array}{l} (pk, msk) \leftarrow \text{FE.Setup}(1^\lambda) \\ sk_C \leftarrow \text{FE.KeyGen}(msk, C) : C(m) = \text{FE.Dec}(sk_C, c) \\ c \leftarrow \text{FE.Enc}(pk, m) \end{array} \right] = 1$$

- **Selective Security:** For every nuPPT A there exists a negligible function μ such that for every $\lambda \in \mathbb{N}$, every circuit $C \in C_\lambda$ with input length n and pair of messages $m_0, m_1 \in \{0, 1\}^n$ such that $C(m_0) = C(m_1)$ we have that $|\Pr[A(\mathcal{D}_0) = 1] - \Pr[A(\mathcal{D}_1) = 1]| \leq \mu(\lambda)$ where

$$\mathcal{D}_b = \Pr \left[\begin{array}{l} (pk, msk) \leftarrow \text{FE.Setup}(1^\lambda) \\ sk_C \leftarrow \text{FE.KeyGen}(msk, C) : (pk, sk_C, c_b) \\ c_b \leftarrow \text{FE.Enc}(pk, m_b) \end{array} \right]$$

We say the scheme has sub-exponential security if there exists a constant ϵ such that for every λ , every 2^{λ^ϵ} -size adversary A , $|\Pr[A(\mathcal{D}_0) = 1] - \Pr[A(\mathcal{D}_1) = 1]| \leq 1/2^{\lambda^\epsilon}$ where \mathcal{D}_b is defined above.

We recall the definition of compactness and succinctness for functional encryption schemes, as defined in [22, 3].

Definition 29 (Compact Functional Encryption). We say a functional encryption scheme for a class of circuits $\{C_\lambda\}$ is compact if for every $\lambda \in \mathbb{N}$, $pk \leftarrow \text{FE.Setup}(1^\lambda)$ and $m \in \{0, 1\}^*$ we have that $\text{Time}(\text{FE.Enc}(pk, m)) = \text{poly}(\lambda, |m|, \log s)$ where $s =$

$$\max_{C \in \mathcal{C}_\lambda} |C|.$$

We say the scheme has sub-linear compactness if the running time of FE.Enc is bounded as $\text{Time}(\text{FE.Enc}(pk, m)) = \text{poly}(\lambda, |m|) \cdot s^{1-\epsilon}$ where $\epsilon > 0$.

Definition 30 (Succinct Functional Encryption). *A compact functional encryption scheme for a class of circuits that output only a single bit is called a succinct functional encryption scheme.*

Theorem 19 ([73]). *Assuming (sub-exponentially secure) LWE, there exists a (sub-exponentially secure) succinct functional encryption scheme for NC^1 .*

We note that [73] do not explicitly consider sub-exponentially secure succinct functional encryption, but their construction satisfies it (assuming sub-exponentially secure LWE). Additionally, we have the following bootstrapping theorem:

Theorem 20 ([63, 2, 3]). *Assuming the existence of symmetric-key encryption with decryption in NC^1 (resp. sub-exponentially secure) and succinct FE for NC^1 (resp. sub-exponentially secure), there exists succinct FE for P/poly (resp. sub-exponentially secure).*

In this paper, we will consider a weaker compactness notion, where only the ciphertext size (but not the encryption time) is sub-linear in the output length of the function being evaluated.

Definition 31 (Weakly Sublinear Compact Functional Encryption). *We say a functional encryption scheme for a class of circuits $\{\mathcal{C}_\lambda\}$ is weakly sublinear compact if there exists $\epsilon > 0$ such that for every $\lambda \in \mathbb{N}$, $pk \leftarrow \text{FE.Setup}(1^\lambda)$ and $m \in \{0, 1\}^*$*

we have that

$$\begin{aligned}\text{Time}_{\text{FE.Enc}}(pk, m) &= \text{poly}(\lambda, |m|, s) \\ \text{outlen}_{\text{FE.Enc}}(pk, m) &= s^{1-\epsilon} \cdot \text{poly}(\lambda, |m|)\end{aligned}$$

where $s = \max_{C \in \mathcal{C}_\lambda} |C|$.

5.2.3 Indistinguishability Obfuscation

We recall the notion of indistinguishability obfuscation ($i\mathcal{O}$).

Definition 32 (Indistinguishability Obfuscator). *A PPT machine $i\mathcal{O}$ is an indistinguishability obfuscator for a circuit class $\{\mathcal{C}_\lambda\}_{\lambda \in \mathbb{N}}$ if the following conditions are satisfied:*

- **Functionality:** *for all security parameters $\lambda \in \mathbb{N}$, for all $C \in \mathcal{C}_\lambda$, for all inputs x , we have that*

$$\Pr[C'(x) = C(x) : C' \leftarrow i\mathcal{O}(C)] = 1 .$$

- **Indistinguishability:** *for any polysize distinguisher \mathcal{D} , there exists a negligible function μ such that the following holds: For all security parameters $\lambda \in \mathbb{N}$, for all pairs of circuits $C_0, C_1 \in \mathcal{C}_\lambda$ of the same size, we have that if $C_0(x) = C_1(x)$ for all inputs x , then*

$$\left| \Pr[\mathcal{D}(i\mathcal{O}(C_0)) = 1] - \Pr[\mathcal{D}(i\mathcal{O}(C_1)) = 1] \right| \leq \mu(\lambda) .$$

We say the scheme has sub-exponential security if there exists a constant ϵ such that for every λ , every 2^{λ^ϵ} -size adversary \mathcal{D} , $|\Pr[\mathcal{D}(i\mathcal{O}(C_0)) = 1] - \Pr[\mathcal{D}(i\mathcal{O}(C_1)) = 1]| \leq 1/2^{\lambda^\epsilon}$.

We recall the following result from [92].

Theorem 21. ([92]) *Assume the existence of sub-exponentially secure LWE. If there exists a weakly sublinear compact functional encryption scheme for \mathbf{P}/poly with sub-exponential security, then there exists a sub-exponentially secure indistinguishability obfuscator for \mathbf{P}/poly .*

5.3 Exponentially-Efficient $i\mathcal{O}$ (\mathbf{XiO})

In this section, we define our new notion of “inefficient” $i\mathcal{O}$, which allows the obfuscator to have running time as long as a brute-force canonicalizer that outputs the entire input-output table of the function, but requires the obfuscated program to be slightly smaller in size than a brute-force canonicalization.

Definition 33 (Exponentially-Efficient Indistinguishability Obfuscation ($i\mathcal{O}$)). *A machine \mathbf{XiO} is an weak indistinguishability obfuscator for a circuit class $\{C_\lambda\}_{\lambda \in \mathbb{N}}$ if the following conditions are satisfied:*

- **Functionality:** *for all security parameters $\lambda \in \mathbb{N}$, for all $C \in C_\lambda$, for all inputs x , we have that*

$$\Pr[C' \leftarrow \mathbf{XiO}(1^\lambda, C) : C'(x) = C(x)] = 1 .$$

- **Indistinguishability:** *for any nuPPT distinguisher A , there exists a negligible function μ such that the following holds: For all security parameters $\lambda \in \mathbb{N}$, for all pairs of circuits $C_0, C_1 \in C_\lambda$ of the same size, we have that if $C_0(x) = C_1(x)$ for all inputs x , then*

$$|\Pr[A(\mathbf{XiO}(1^\lambda, C_0)) = 1] - \Pr[A(\mathbf{XiO}(1^\lambda, C_1)) = 1]| \leq \mu(\lambda)$$

We say the scheme has sub-exponential security if there exists a constant ϵ such that for every λ , every 2^{λ^ϵ} -size adversary A ,

$$|\Pr[A(\mathbf{XiO}(1^\lambda, C_0)) = 1] - \Pr[A(\mathbf{XiO}(1^\lambda, C_1)) = 1]| \leq 1/2^{\lambda^\epsilon}$$

- **Non-trivial Efficiency:** There exists a constant $\epsilon > 0$ such that for any security parameter $\lambda \in \mathbb{N}$, circuit $C \in C_\lambda$ with input length n and $C' \in \mathbf{XiO}(1^\lambda, C)$, we have that

$$\text{Time}_{\mathbf{XiO}}(1^\lambda, C) = \text{poly}(\lambda, |C| \cdot 2^n)$$

$$\text{outlen}_{\mathbf{XiO}}(1^\lambda, C) = \text{poly}(\lambda, |C|) \cdot 2^{n(1-\epsilon)}$$

Remark 3. (Circuits with logarithmic input length) Note that if we want the obfuscation to be efficient (i.e., polynomial-time in λ and the size of the circuit to be obfuscated), then the above definition is only meaningful when the class of circuits C_λ has input length $O(\log \lambda)$. Our results in this paper hold assuming Exponentially-Efficient iO for such classes.

Remark 4. (Exponentially-Efficient iO in the preprocessing model and comparison with Compact Functional Encryption) We can consider further a relaxation of the running-time requirement of the obfuscator. The obfuscator may first perform a long “pre-processing” step (without having seen the program to be obfuscated), taking time $\text{poly}(\lambda, s, 2^n)$ (where s is the size bound on circuits to be obfuscated), and outputting a (potentially long) pre-processing public-key O_{pk} . The actual obfuscation then takes O_{pk} , and the circuit C as inputs, runs in time $\text{poly}(\lambda, s, 2^n)$ and outputs an obfuscated program of size $\text{poly}(\lambda, s) \cdot 2^{n(1-\epsilon)}$, and then the evaluation of the obfuscated program may finally also access the public-key O_{pk} . All our results also apply to this relaxed notion of exponentially efficient iO .

Additionally, we note that weakly sublinear compact FE directly implies this notion as follows: pre-processing public key O_{pk} (generated in the pre-processing step) is the public key pk for the FE and the functional secret key sk_{FT} corresponding to a function table generator program that takes as input a circuit and outputs the function table of it; the obfuscation of a circuit C is an encryption of the circuit C (w.r.t., the FE public key pk), and evaluation of the obfuscated code uses the functional secret key sk_{FT} inside O_{pk} to compute the function table of C and selects the appropriate output. Sub-linear compactness of the functional encryption scheme implies the obfuscator has exponential efficiency.

Remark 5. (Comparison with Compact Randomized Encoding for Turing machines) [3] and [92] study a notion of compact randomized encodings [83, 5]. Roughly speaking, a randomized encoding (RE) is a method for encoding a Turing Machine Π , an input x and a running-time bound T , into a randomized encoding $\widehat{\Pi}(x)$ from which $\Pi(x)$ can be efficiently decoded; furthermore the encodings does not leak anything more about Π and x than what can be (inefficiently) deduced from just the output $\Pi(x)$ (truncated at T steps).³ A randomized encodings is compact (resp. sublinearly compact) if the encoding time is poly-logarithmic (resp sublinear) in T (and polynomial in the size of Π and x). We note that sublinear compact RE directly implies **XiO** as follows: to obfuscate a circuit C , compute an encoding \widehat{FT}_C of the function table generator Turing machine FT_C that has the circuit C hardcoded (i.e., FT_C takes no inputs and simply computes the function table of C); evaluation of the obfuscation on an input i simply decodes the encoding \widehat{FT}_C picks out the i th output. Sublinear compactness of the RE implies that the obfuscator is exponentially-efficient.

In fact, the above methods extend to show that (sublinearly) compact RE implies

³Or equivalently, for any two programs Π_1, Π_2 and inputs x_1, x_2 such that $\Pi_1(x_1) = \Pi_2(x_1)$, a randomized encoding of Π_1, x is indistinguishable from an encoding of Π_2, x_2 .

a notion of **XiO** for Turing machines. We note that Turing machine obfuscation is a significantly harder task than circuit obfuscation (indeed, all known construction of Turing machine obfuscators first go through circuit obfuscation). We also point out that whereas (subexponentially-secure) *iO* for circuits is known to imply *iO* for Turing machine [19, 43, 90], these techniques do not apply in the regime of programs with short input.

5.4 *iO* from **XiO**

In this section, we show how to achieve (full-fledged) *iO* from **XiO**.

5.4.1 Weakly Sublinear Compact FE from Succinct FE and **XiO**

We first give our construction of weakly sublinear compact FE from succinct FE and **XiO** for circuits with input-size $O(\log(\lambda))$. At a high-level, our idea is to have the ciphertext for the FE scheme be **XiO** of a circuit that, on input i , generates a succinct FE encryption of (m, i) . The secret key corresponding to C consists of a single key for the succinct FE scheme, that, given a ciphertext encrypting (m, i) , computes the i th output bit of $C(m)$.

Let F be a puncturable pseudorandom function, **XiO** be a exponentially efficient indistinguishability obfuscator for $\text{P}^{O(\log \lambda)}/\text{poly}$ and sFE be a succinct functional encryption scheme (resp. with sub-exponential security) for an appropriate class of circuits that includes C' defined below.. We define a compact functional encryption scheme FE for a class of poly-size circuits $\{C_\lambda\}$ as follows:

$(msk, pk) \leftarrow \text{FE.Setup}(1^\lambda)$: FE.Setup is identical to sFE.Setup and has the same output.

$c \leftarrow \text{FE.Enc}(pk, m)$: FE.Enc samples a PRF key $K \leftarrow \text{F.Key}(1^\lambda)$ and outputs $\mathbf{XiO}(1^\lambda, G[pk, K])$ where $G[pk, K]$ is a circuit with input length $n = \log s$ where $s = \max_{C \in \mathcal{C}_\lambda} \text{outlen}(C)$, defined as follows:

$$G[pk, K](i) = \text{sFE.Enc}(pk, (m, i); \text{F.Eval}(K, i))$$

G is padded to be the same size as another circuit G'' , which we will define later in the security proof. Both G and G'' will ultimately have size $S = \text{poly}(\lambda, |m|, \log s)$ where $s = \max_{C \in \mathcal{C}_\lambda} |C|$.

$sk_C \leftarrow \text{FE.KeyGen}(msk, C)$: FE.KeyGen outputs $\text{sFE.KeyGen}(msk, C')$ where C' on input (m, i) outputs the i^{th} bit of $C(m)$, or outputs \perp if i is greater than the output length of C .

$y \leftarrow \text{FE.Dec}(sk_C, c)$: FE.Dec runs $c_i \leftarrow G[pk, K](i)$ and $y_i \leftarrow \text{sFE.Dec}(sk_C, c_i)$ for every i and outputs y_1, \dots, y_{2^n} .

Let $\{C'_\lambda\}$ be a class of circuits that includes C' as defined above for every $C \in \mathcal{C}_\lambda$.

Theorem 22. *Assuming F is a pseudorandom function (resp. with subexponential security), \mathbf{XiO} is an exponentially efficient indistinguishability obfuscator for $\text{P}^{\text{O}(\log \lambda)}/\text{poly}$ (resp. with subexponential security) and sFE is a succinct functional encryption scheme for $\{C'_\lambda\}$ (resp. with subexponential security), we have that FE as defined above is a functional encryption scheme for $\{C_\lambda\}$ with weakly sub-linear compactness (resp. and with subexponential security).*

Proof. We first show weak sublinear compactness of FE . Consider any $\lambda, C \in \mathcal{C}_\lambda$, message m , $pk \in \text{FE.Setup}(1^\lambda)$ and PRF key $K \in \{0, 1\}^\lambda$. $\text{Time}(\text{FE.Enc}(pk, m))$

is the time \mathbf{XiO} takes to obfuscate the circuit $G[pk, K]$, which is of size $S = \text{poly}(\lambda, |m|, \log s)$ where $s = \max_{C \in \mathcal{C}_\lambda} |C|$. Hence we have that

$$\begin{aligned} \text{Time}_{\mathbf{XiO}}(1^\lambda, G[pk, K]) &= \text{poly}(\lambda, |m|, \log s, \cdot 2^n) \leq \text{poly}(\lambda, |m|, s) \\ \text{outlen}_{\mathbf{XiO}}(1^\lambda, G[pk, K]) &= \text{poly}(\lambda, |m|, \log s) \cdot 2^{n(1-\epsilon)} \leq \text{poly}(\lambda, |m|) \cdot s^{1-\epsilon'} \end{aligned}$$

where ϵ' is a constant with $0 < \epsilon' < \epsilon$.

Next we show the selective security of FE. The proof proceeds by a hybrid argument where in each hybrid distribution, the circuit being obfuscated, on input i , produces ciphertexts of m_1 when i is less than a “threshold”, and ciphertexts of m_0 otherwise. Indistinguishability of neighboring hybrids is shown using the “punctured programming” technique of [116], as was done in [46] for constructing iO for probabilistic functions. This technique is also used extensively in other applications of iO , eg., [19], [43], [90] and more.

Assume for contradiction there exists a nuPPT A and polynomial p such that for sufficiently large λ , circuit $C \in \mathcal{C}_\lambda$ and messages m_0, m_1 such that $C(m_0) = C(m_1)$, A distinguishes \mathcal{D}_0 and \mathcal{D}_1 as defined in Definition 28 with advantage $1/p(\lambda)$. For $j \in [l]$, we define the j^{th} hybrid distribution H_j as follows:

$$H_j = \left(\begin{array}{l} (msk, pk) \leftarrow \text{FE.Setup}(1^\lambda) \\ K \leftarrow \{0, 1\}^\lambda \quad : \quad pk, sk_C, \mathbf{XiO}(G'[pk, K, j, m_0, m_1]) \\ sk_C \leftarrow \text{FE.KeyGen}(msk, C) \end{array} \right)$$

where $G'[pk, K, j, m_0, m_1]$, where G' is defined as follows

$$G'[pk, K, j, m_0, m_1](i) = \begin{cases} \text{sFE.Enc}(pk, (m_0, i); F(K, i)) & \text{if } i > j \\ \text{sFE.Enc}(pk, (m_1, i); F(K, i)) & \text{if } i \leq j \end{cases}$$

We also require G' to be padded to be of the same size S as $G[pk, K, m]$.

We consider the hybrid sequence $\mathcal{D}_0, H_1, \dots, H_l, \mathcal{D}_1$. By a hybrid argument, there exists a pair of neighboring hybrids in this sequence such that A distinguishes the pair with probability $\frac{1}{p(\lambda) \cdot (l+2)} = \frac{1}{\text{poly}(\lambda)}$. We show a contradiction by proving that each pair of neighboring hybrids is computationally indistinguishable.

We first note that \mathcal{D}_0 is indistinguishable from H_0 . This follows by observing that $G'[pk, K, 0, m_0, m_1]$ is functionally identical to $G[pk, K, m_0]$, and applying the security of **XiO**. The same argument also shows that H_l is indistinguishable from \mathcal{D}_1 .

Next, we show H_{j^*} and H_{j^*+1} are indistinguishable for each $j^* \in [l]$. Define hybrid distribution H'_0 which is identical to H_{j^*} except that **XiO** obfuscates a different circuit $G''[pk, K_{j^*}, j^*, m_0, m_1, c]$ where $K_{j^*} \leftarrow \text{Punc}(\lambda, j^*)$ and $c \leftarrow \text{sFE.Enc}(pk, (m_0, j^*); R)$ using uniformly sampled randomness R . G'' on input i has the same behavior as G' except $i = j^*$, where it outputs the hardcoded ciphertext c . The padding parameter S is defined as the size of G'' , which is $\text{poly}(\lambda, |m|, \log s)$. By the “punctured programming” technique of Sahai-Waters [116], which relies on the security of the obfuscator **XiO** and puncturable PRF F , it follows that for sufficiently large λ , A distinguishes between H_{j^*} and H'_0 with negligible probability.

The puncturing programming technique itself works in two hybrid steps:

- First the circuit G' is modified on input j^* to output a hardcoded value $\text{sFE.Enc}(pk, (m_0, j^*); F(K, j^*))$, which is the same ciphertext G' previously computed. Also, the PRF key in G' is modified to be punctured on input j^* . Since this doesn't change the functionality of the circuit, indistinguishable.

bility follows from the security of **XiO**.

- Second, the hardcoded ciphertext is modified to be generated from real randomness R , and indistinguishability follows from the security of the puncturable PRF.

Next, we define hybrid distribution H'_1 which is identical to H'_0 except that the hardcoded ciphertext c is generated as $\text{sFE.Enc}(pk, (m_1, j^*); R)$ for uniformly sampled randomness R . Since $C(m_0)$ is identical to $C(m_1)$, from the security of sFE, A distinguishes H'_0 and H'_1 with negligible probability.

Finally, note that H'_1 and H_{j^*+1} differ in the same way H'_0 and H_{j^*} do, and are hence indistinguishable by a similar argument. Hence A distinguishes H_{j^*} and H_{j^*+1} with negligible probability and we have a contradiction. This completes the proof.

We note that the proof above is described in terms of computational indistinguishability, but in fact also can be applied to show that FE is subexponentially-secure, if both **XiO** and sFE are subexponentially secure. \square

Theorem 23. *Assuming sub-exponentially hard LWE, if there exists a sub-exponentially-secure exponentially efficient indistinguishability obfuscator for P/poly then there exists an indistinguishability obfuscator for P/poly with sub-exponential security.*

Proof. By Theorem 19 and Theorem 7, assuming subexponentially secure LWE, there exists a succinct functional encryption scheme for P/poly that is subexponentially-secure. Using this with a subexponentially-secure exponentially efficient indistinguishability obfuscator, by Theorem 22, we get weakly

sublinear compact function encryption for P/poly with sub-exponential selective security. Together with Theorem 21, this gives us iO for P/poly. \square

Remark 6. (*XiO for NC^1 suffices*) Note that we can also achieve the above results assuming XiO for only NC^1 instead of P/poly, with the caveat that we must additionally assume the existence of puncturable PRFs in NC^1 . To do so, we modify our above construction, so that instead of obfuscating the circuit G , we instead obfuscate a different circuit H that generates a “garbling” [118] of G . Since H lies in NC^1 , we only need XiO for NC^1 .

5.5 Comparison with [3]

In this section we briefly describe the related result by [3] and compare it with our result. [3] show how to construct a Compact Functional Encryption scheme from a Succinct Functional Encryption scheme and Compact Randomized Encodings for Turing machines. The rough idea is as follows: the compact functional secret key for a function f is a sequence of ℓ independent succinct functional secret keys where ℓ is the output length of f . The i^{th} succinct functional secret key corresponds to the function that outputs the i^{th} bit of f . The compact functional ciphertext for a message m is the randomized encoding of a machine Π that takes no input and when run, outputs $\{Enc(pk_i, m)\}_{i \in [\ell]}$ where pk_i is the public key corresponding to the i^{th} instance of the succinct functional scheme (these instances are generated using a PRF, hence the description size of Π is independent of ℓ). The compactness of the functional encryption scheme follows from the compactness of the randomized encoding scheme.

Note that the above result necessarily requires the computation being en-

coded be represented as a Turing machine, since the description size is required to be independent of the output length. In contrast we are able to rely on **XiO** for *circuits*, which is significantly weaker (see Remark 5).

BIBLIOGRAPHY

- [1] Prabhanjan Ananth, Dan Boneh, Sanjam Garg, Amit Sahai, and Mark Zhandry. Differing-inputs obfuscation and applications. *IACR Cryptology ePrint Archive*, 2013:689, 2013.
- [2] Prabhanjan Ananth, Zvika Brakerski, Gil Segev, and Vinod Vaikuntanathan. The trojan method in functional encryption: From selective to adaptive security. Technical report, generically. *Cryptology ePrint Archive*, Report 2014/917, 2014.
- [3] Prabhanjan Ananth and Abhishek Jain. Indistinguishability obfuscation from compact functional encryption. *IACR Cryptology ePrint Archive*, 2015:173, 2015.
- [4] Benny Applebaum. Randomly encoding functions: A new cryptographic paradigm - (invited talk). In *Information Theoretic Security - 5th International Conference, ICITS 2011, Amsterdam, The Netherlands, May 21-24, 2011. Proceedings*, pages 25–31, 2011.
- [5] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Cryptography in nc^0 . In *FOCS*, pages 166–175, 2004.
- [6] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Computationally private randomizing polynomials and their applications. *Computational Complexity*, 15(2):115–162, 2006.
- [7] Boaz Barak. How to go beyond the black-box simulation barrier. In *FOCS*, volume 0, pages 106–115, 2001.
- [8] Boaz Barak. *Non-black-box techniques in cryptography*. PhD thesis, Citeseer, 2004.
- [9] Boaz Barak, Sanjam Garg, Yael Tauman Kalai, Omer Paneth, and Amit Sahai. Protecting obfuscation against algebraic attacks. *Cryptology ePrint Archive*, Report 2013/631, 2013.
- [10] Boaz Barak and Oded Goldreich. Universal arguments and their applications. In *Computational Complexity*, pages 162–171, 2002.
- [11] Boaz Barak, Oded Goldreich, Shafi Goldwasser, and Yehuda Lindell.

- Resettably-sound zero-knowledge and its applications. In *FOCS*, pages 116–125, 2001.
- [12] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In *Advances in Cryptology CRYPTO 2001*, pages 1–18. Springer, 2001.
- [13] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. *J. ACM*, 59(2):6, 2012.
- [14] David A. Mix Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in nc^1 . In *STOC*, pages 1–5, 1986.
- [15] Mihir Bellare, Igors Stepanovs, and Stefano Tessaro. Poly-many hardcore bits for any one-way function. *Cryptology ePrint Archive*, Report 2013/873, 2013. <http://eprint.iacr.org/>.
- [16] Nir Bitansky and Ran Canetti. On strong simulation and composable point obfuscation. In *CRYPTO*, pages 520–537, 2010.
- [17] Nir Bitansky, Ran Canetti, Yael Kalai, and Omer Paneth. Virtual-grey-box obfuscation from general circuits. In *Advances in Cryptology CRYPTO 2014*, 2014.
- [18] Nir Bitansky, Ran Canetti, Omer Paneth, and Alon Rosen. Indistinguishability obfuscation vs. auxiliary-input extractable functions: One must fall. *IACR Cryptology ePrint Archive*, 2013:641, 2013.
- [19] Nir Bitansky, Sanjam Garg, Huijia Lin, Rafael Pass, and Sidharth Telang. Succinct randomized encodings and their applications. *IACR Cryptology ePrint Archive*, 2015:356, 2015.
- [20] Nir Bitansky and Omer Paneth. From the impossibility of obfuscation to a new non-black-box simulation technique. In *FOCS*, 2012.
- [21] Nir Bitansky and Omer Paneth. Zaps and non-interactive witness indistinguishability from indistinguishability obfuscation. In *Theory of Cryptography - 12th Theory of Cryptography Conference, TCC 2015, Warsaw, Poland, March 23-25, 2015, Proceedings, Part II*, pages 401–427, 2015.

- [22] Nir Bitansky and Vinod Vaikuntanathan. Indistinguishability obfuscation from functional encryption. *IACR Cryptology ePrint Archive*, 2015:163, 2015.
- [23] M. Blum. How to prove a theorem so no one else can claim it. *Proc. of the International Congress of Mathematicians*, pages 1444–1451, 1986.
- [24] Dan Boneh, editor. *Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003, Proceedings*, volume 2729 of *Lecture Notes in Computer Science*. Springer, 2003.
- [25] Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical identity based encryption with constant size ciphertext. In *Advances in Cryptology EUROCRYPT 2005*, pages 440–456. 2005.
- [26] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In Yuval Ishai, editor, *Theory of Cryptography - 8th Theory of Cryptography Conference, TCC 2011, Providence, RI, USA, March 28-30, 2011. Proceedings*, volume 6597 of *Lecture Notes in Computer Science*, pages 253–273. Springer, 2011.
- [27] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: a new vision for public-key cryptography. *Commun. ACM*, 55(11):56–64, 2012.
- [28] Dan Boneh and Alice Silverberg. Applications of multilinear forms to cryptography. *Contemporary Mathematics*, 324(1):71–90, 2003.
- [29] Dan Boneh, David J Wu, and Joe Zimmerman. Immunizing multilinear maps against zeroizing attacks. *IACR Cryptology ePrint Archive*, 2014:930, 2014.
- [30] Dan Boneh and Mark Zhandry. Multiparty key exchange, efficient traitor tracing, and more from indistinguishability obfuscation. In *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part I*, pages 480–499, 2014.
- [31] Dan Boneh and Mark Zhandry. Multiparty key exchange, efficient traitor tracing, and more from indistinguishability obfuscation. In *Advances in Cryptology CRYPTO 2014*, 2014.

- [32] Elette Boyle, Kai-Min Chung, and Rafael Pass. On extractability obfuscation. In *TCC*, pages 52–73, 2014.
- [33] Elette Boyle and Rafael Pass. Limits of extractability assumptions with distributional auxiliary input. 2013.
- [34] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. In *ITCS*, pages 309–325, 2012.
- [35] Zvika Brakerski, Ilan Komargodski, and Gil Segev. From single-input to multi-input functional encryption in the private-key setting. *IACR Cryptology ePrint Archive*, 2015:158, 2015.
- [36] Zvika Brakerski and Guy N. Rothblum. Virtual black-box obfuscation for all circuits via generic graded encoding. In *TCC*, pages 1–25, 2014.
- [37] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) lwe. In *FOCS*, pages 97–106, 2011.
- [38] Christina Brzuska and Arno Mittelbach. Indistinguishability obfuscation versus point obfuscation with auxiliary input. *Cryptology ePrint Archive*, Report 2014/405, 2014. <http://eprint.iacr.org/>.
- [39] Christina Brzuska and Arno Mittelbach. Using indistinguishability obfuscation via uces. *Cryptology ePrint Archive*, Report 2014/381, 2014. <http://eprint.iacr.org/>.
- [40] Ran Canetti. Towards realizing random oracles: Hash functions that hide all partial information. In *Advances in Cryptology CRYPTO 1997*, pages 455–469, 1997.
- [41] Ran Canetti, Oded Goldreich, Shafi Goldwasser, and Silvio Micali. Resettable zero-knowledge (extended abstract). In *STOC*, pages 235–244, 2000.
- [42] Ran Canetti, Oded Goldreich, and Shai Halevi. On the random-oracle methodology as applied to length-restricted signature schemes. In *TCC*, pages 40–57, 2004.
- [43] Ran Canetti, Justin Holmgren, Abhishek Jain, and Vinod Vaikuntanathan. Indistinguishability obfuscation of iterated circuits and RAM programs. *IACR Cryptology ePrint Archive*, 2014:769, 2014.

- [44] Ran Canetti, Yael Tauman Kalai, and Omer Paneth. On obfuscation with random oracles. In *Theory of Cryptography - 12th Theory of Cryptography Conference, TCC 2015, Warsaw, Poland, March 23-25, 2015, Proceedings, Part II*, pages 456–467, 2015.
- [45] Ran Canetti, Joe Kilian, Erez Petrank, and Alon Rosen. Black-box concurrent zero-knowledge requires $\tilde{\omega}(\log n)$ rounds. In *STOC*, pages 570–579, 2001.
- [46] Ran Canetti, Huijia Lin, Stefano Tessaro, and Vinod Vaikuntanathan. Obfuscation of probabilistic circuits and applications. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *Theory of Cryptography*, volume 9015 of *Lecture Notes in Computer Science*, pages 468–497. Springer Berlin Heidelberg, 2015.
- [47] Jung Hee Cheon, Kyoohyung Han, Changmin Lee, Hansol Ryu, and Damien Stehlé. Cryptanalysis of the multilinear map over the integers. In *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I*, pages 3–12, 2015.
- [48] Kai-Min Chung, Rafail Ostrovsky, Rafael Pass, and Ivan Visconti. Simultaneous resettability from one-way functions. 2013.
- [49] Jean-Sébastien Coron, Tancrede Lepoint, and Mehdi Tibouchi. Practical multilinear maps over the integers. In *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I*, pages 476–493, 2013.
- [50] Jean-Sébastien Coron, Tancrede Lepoint, and Mehdi Tibouchi. Cryptanalysis of two candidate fixes of multilinear maps over the integers. 2014.
- [51] Jean-Sébastien Coron, Tancrede Lepoint, and Mehdi Tibouchi. New multilinear maps over the integers. In *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I*, pages 267–286, 2015.
- [52] 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.
- [53] Uriel Feige and Adi Shamir. Witness indistinguishable and witness hiding protocols. In *STOC*, pages 416–426, 1990.

- [54] Sanjam Garg, Craig Gentry, and Shai Halevi. Candidate multilinear maps from ideal lattices. In *Advances in Cryptology–EUROCRYPT 2013*, pages 1–17. Springer, 2013.
- [55] Sanjam Garg, Craig Gentry, Shai Halevi, and Mariana Raykova. Two-round secure MPC from indistinguishability obfuscation. In *Theory of Cryptography - 11th Theory of Cryptography Conference, TCC 2014, San Diego, CA, USA, February 24-26, 2014. Proceedings*, pages 74–94, 2014.
- [56] Sanjam Garg, Craig Gentry, Shai Halevi, and Mariana Raykova. Two-round secure mpc from indistinguishability obfuscation. In *TCC*, pages 74–94, 2014.
- [57] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. *Proc. of FOCS 2013*, 2013.
- [58] Sanjam Garg, Craig Gentry, Shai Halevi, and Daniel Wichs. On the implausibility of differing-inputs obfuscation and extractable witness encryption with auxiliary input. In *CRYPTO’14*, 2013.
- [59] Sanjam Garg, Craig Gentry, Amit Sahai, and Brent Waters. Witness encryption and its applications. In *Proceedings of the 45th Annual ACM Symposium on Theory of Computing, STOC ’13*, pages 467–476, 2013.
- [60] Craig Gentry. *A fully homomorphic encryption scheme*. PhD thesis, Stanford University, 2009.
- [61] Craig Gentry, Sergey Gorbunov, and Shai Halevi. Graph-induced multilinear maps from lattices. In *Theory of Cryptography - 12th Theory of Cryptography Conference, TCC 2015, Warsaw, Poland, March 23-25, 2015, Proceedings, Part II*, pages 498–527, 2015.
- [62] Craig Gentry, Shai Halevi, Hemanta K Maji, and Amit Sahai. Zeroizing without zeroes: Cryptanalyzing multilinear maps without encodings of zero. *IACR Cryptology ePrint Archive*, 2014:929, 2014.
- [63] Craig Gentry, Shai Halevi, Mariana Raykova, and Daniel Wichs. Outsourcing private ram computation. In *Foundations of Computer Science (FOCS), 2014 IEEE 55th Annual Symposium on*, pages 404–413. IEEE, 2014.

- [64] Craig Gentry, Allison Lewko, Amit Sahai, and Brent Waters. Indistinguishability obfuscation from the multilinear subgroup elimination assumption. *Cryptology ePrint Archive*, Report 2014/309, 2014. <http://eprint.iacr.org/>.
- [65] Craig Gentry, Allison Lewko, and Brent Waters. Witness encryption from instance independent assumptions. In *Advances in Cryptology—CRYPTO 2014*, pages 426–443. Springer, 2014.
- [66] Oded Goldreich. *Foundations of Cryptography — Basic Tools*. Cambridge University Press, 2001.
- [67] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *J. ACM*, 33(4):792–807, 1986.
- [68] Oded Goldreich and Ariel Kahan. How to construct constant-round zero-knowledge proof systems for NP. *Journal of Cryptology*, 9(3):167–190, 1996.
- [69] Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. *J. ACM*, 38(3):690–728, 1991.
- [70] Oded Goldreich and Yair Oren. Definitions and properties of zero-knowledge proof systems. *Journal of Cryptology*, 7:1–32, 1994.
- [71] Shafi Goldwasser, S. Dov Gordon, Vipul Goyal, Abhishek Jain, Jonathan Katz, Feng-Hao Liu, Amit Sahai, Elaine Shi, and Hong-Sheng Zhou. Multi-input functional encryption. In *EUROCRYPT*, pages 578–602, 2014.
- [72] Shafi Goldwasser and Yael Tauman Kalai. On the impossibility of obfuscation with auxiliary input. In *46th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2005), 23-25 October 2005, Pittsburgh, PA, USA, Proceedings*, pages 553–562, 2005.
- [73] Shafi Goldwasser, Yael Tauman Kalai, Raluca A. Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich. Reusable garbled circuits and succinct functional encryption. In *Symposium on Theory of Computing Conference, STOC’13, Palo Alto, CA, USA, June 1-4, 2013*, pages 555–564, 2013.
- [74] Shafi Goldwasser, Yael Tauman Kalai, Raluca A. Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich. Reusable garbled circuits and succinct

- functional encryption. In *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 555–564, 2013.
- [75] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *J. Comput. Syst. Sci.*, 28(2):270–299, 1984.
- [76] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18(1):186–208, 1989.
- [77] Shafi Goldwasser and Guy Rothblum. On best-possible obfuscation. In *Theory of Cryptography*, volume 4392, pages 194–213. 2007.
- [78] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Attribute-based encryption for circuits. In *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 545–554, 2013.
- [79] Satoshi Hada. Zero-knowledge and code obfuscation. In *Advances in Cryptology—ASIACRYPT 2000*, pages 443–457. Springer, 2000.
- [80] Johan Hstad, Russell Impagliazzo, Leonid Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28:12–24, 1999.
- [81] Susan Hohenberger, Amit Sahai, and Brent Waters. Replacing a random oracle: Full domain hash from indistinguishability obfuscation. In *EUROCRYPT*, pages 201–220, 2014.
- [82] Yuval Ishai and Eyal Kushilevitz. Randomizing polynomials: A new representation with applications to round-efficient secure computation. In *41st Annual Symposium on Foundations of Computer Science, FOCS 2000, 12-14 November 2000, Redondo Beach, California, USA*, pages 294–304, 2000.
- [83] Yuval Ishai and Eyal Kushilevitz. Perfect constant-round secure computation via perfect randomizing polynomials. In *ICALP*, pages 244–256, 2002.
- [84] Yuval Ishai and Eyal Kushilevitz. Perfect constant-round secure computation via perfect randomizing polynomials. In *Automata, Languages and Programming, 29th International Colloquium, ICALP 2002, Malaga, Spain, July 8-13, 2002, Proceedings*, pages 244–256, 2002.

- [85] Yuval Ishai, Omkant Pandey, and Amit Sahai. Public-coin differing-inputs obfuscation and its applications. In *Theory of Cryptography - 12th Theory of Cryptography Conference, TCC 2015, Warsaw, Poland, March 23-25, 2015, Proceedings, Part II*, pages 668–697, 2015.
- [86] Joe Kilian. Founding cryptography on oblivious transfer. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*, pages 20–31. ACM, 1988.
- [87] Joe Kilian. A note on efficient zero-knowledge proofs and arguments. In *Proceedings of the twenty-fourth annual ACM symposium on Theory of computing*, pages 723–732. ACM, 1992.
- [88] Ilan Komargodski, Tal Moran, Moni Naor, Rafael Pass, Alon Rosen, and Eylon Yogev. One-way functions and (im)perfect obfuscation. *Cryptology ePrint Archive*, Report 2014/347, 2014. <http://eprint.iacr.org/>.
- [89] Ilan Komargodski, Moni Naor, and Eylon Yogev. Secret-sharing for np. *Cryptology ePrint Archive*, Report 2014/213, 2014. <http://eprint.iacr.org/>.
- [90] Venkata Koppula, Allison Bishop Lewko, and Brent Waters. Indistinguishability obfuscation for turing machines with unbounded memory. Technical report, *Cryptology ePrint Archive*, Report 2014/925, 2014. http://eprint.iacr.org, 2014.
- [91] Huijia Lin and Rafael Pass. Constant-round non-malleable commitments from any one-way function. In *STOC*, pages 705–714, 2011.
- [92] Huijia Lin, Rafael Pass, Karn Seth, and Sidharth Telang. Output-compressing randomized encodings and applications. 2015.
- [93] Yehuda Lindell. Lower bounds for concurrent self composition. In *TCC*, pages 203–222, 2004.
- [94] Mohammad Mahmoody, Ameer Mohammed, and Soheil Nematihaji. More on impossibility of virtual black-box obfuscation in idealized models. *IACR Cryptology ePrint Archive*, 2015:632, 2015.
- [95] Ralph C. Merkle. A certified digital signature. In *CRYPTO*, pages 218–238, 1989.

- [96] Silvio Micali. Computationally sound proofs. *SIAM Journal on Computing*, 30(4):1253–1298, 2000.
- [97] Brice Minaud and Pierre-Alain Fouque. Cryptanalysis of the new multilinear map over the integers. *Cryptology ePrint Archive, Report 2015/941*, 2015. <http://eprint.iacr.org/>.
- [98] Moni Naor. Bit commitment using pseudorandomness. *Journal of Cryptology*, 4:151–158, 1991.
- [99] Moni Naor. On cryptographic assumptions and challenges. In Boneh [24], pages 96–109.
- [100] Moni Naor. On cryptographic assumptions and challenges. In Boneh [24], pages 96–109.
- [101] Moni Naor and Moti Yung. Universal one-way hash functions and their cryptographic applications. In *STOC*, pages 33–43, 1989.
- [102] Moni Naor and Moti Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *Proceedings of the twenty-second annual ACM symposium on Theory of computing*, pages 427–437, 1990.
- [103] Rafail Ostrovsky and Avi Wigderson. One-way functions are essential for non-trivial zero-knowledge. In *Theory and Computing Systems, 1993*, pages 3–17, 1993.
- [104] Rafael Pass. Bounded-concurrent secure multi-party computation with a dishonest majority. In *STOC*, pages 232–241, New York, NY, USA, 2004. ACM.
- [105] Rafael Pass and Alon Rosen. New and improved constructions of non-malleable cryptographic protocols. In *STOC*, pages 533–542, 2005.
- [106] Rafael Pass, Karn Seth, and Sidharth Telang. Indistinguishability obfuscation from semantically-secure multilinear encodings. *Cryptology ePrint Archive, Report 2013/781*, 2013.
- [107] Rafael Pass, Karn Seth, and Sidharth Telang. Indistinguishability obfuscation from semantically-secure multilinear encodings. In *Advances in Cryptology—CRYPTO 2014*, pages 500–517. Springer, 2014.

- [108] Rafael Pass and Abhi Shelat. Impossibility of VBB obfuscation with ideal constant-degree graded encodings. *IACR Cryptology ePrint Archive*, 2015:383, 2015.
- [109] Rafael Pass, Wei-Lung Dustin Tseng, and Douglas Wikström. On the composition of public-coin zero-knowledge protocols. In *CRYPTO '09*, pages 160–176, 2009.
- [110] 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.
- [111] Manoj Prabhakaran, Alon Rosen, and Amit Sahai. Concurrent zero knowledge with logarithmic round-complexity. In *FOCS*, pages 366–375, 2002.
- [112] R L Rivest, L Adleman, and M L Dertouzos. On data banks and privacy homomorphisms. *Foundations of Secure Computation, Academia Press*, pages 169–179, 1978.
- [113] John Rompel. One-way functions are necessary and sufficient for secure signatures. In *STOC*, pages 387–394, 1990.
- [114] Alon Rosen. *Concurrent Zero-Knowledge: With Additional Background by Oded Goldreich*. Springer Science & Business Media, 2007.
- [115] Ron D Rothblum. On the circular security of bit-encryption. In *Theory of Cryptography*, pages 579–598. Springer, 2013.
- [116] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: Deniable encryption, and more. *Proc. of STOC 2014*, 2014.
- [117] Andrew C Yao. Theory and application of trapdoor functions. In *Foundations of Computer Science, 1982. SFCS'08. 23rd Annual Symposium on*, pages 80–91. IEEE, 1982.
- [118] Andrew Chi-Chih Yao. Protocols for secure computations (extended abstract). In *23rd Annual Symposium on Foundations of Computer Science, Chicago, Illinois, USA, 3-5 November 1982*, pages 160–164, 1982.