The Computational Complexity of Program Obfuscation

A Dissertation

Presented to

the faculty of the School of Engineering and Applied Science

University of Virginia

---

in partial fulfillment
of the requirements for the degree

Doctor of Philosophy

by

Ameer Mohammed

May 2018

APPROVAL SHEET

This Dissertation
is submitted in partial fulfillment of the requirements
for the degree of
Doctor of Philosophy

Author Signature: 

This Dissertation has been read and approved by the examining committee:

Advisor: Mohammad Mahmoody

Committee Member: David Evans

Committee Member: Jack Davidson

Committee Member: Sanjam Garg

Committee Member: Denis Nekipelov

Committee Member:

Accepted for the School of Engineering and Applied Science:

Craig H. Benson, School of Engineering and Applied Science

May 2018

Abstract

Recent developments in cryptography have given rise to a variety of tools that allow for secure communication and computation of data. In this work we focus on studying one such task called *program obfuscation* which, roughly speaking, hides any information within a program's code without affecting its functionality. In particular, we will study the complexity of the main variant of obfuscation called *indistinguishability obfuscation* (IO), which is itself versatile enough to enable a wide variety of new applications in cryptography, and our main objective is determine a lower bound on the assumptions required for realizing an IO scheme.

The reason behind pursuing this objective of proving such lower bounds for IO (and in general for any other cryptographic primitive) is to understand the complexity of said primitive and to investigate whether it is possible to use well-studied assumptions to base this primitive on. In order to prove these kinds of lower bounds for IO, we need to show that certain assumptions and/or objects are insufficient to achieve it. Such classes of *impossibility* results are proven under the "black-box framework" introduced by Impagliazzo and Rudich in 1989 and later formalized by Reingold, Trevisan, and Vadhan in 2004. However, due to the nature of current IO constructions, the existing techniques for proving impossibility results are not conducive to ruling out the type of "complex" assumptions on which IO stands.

The goal of this thesis is to first develop new techniques and propose extensions to the aforementioned classical black-box framework, allowing us to rule out some natural primitives from being able to construct IO. This then allows us to explore and prove lower bounds on the complexity of obfuscation by showing that certain assumptions and/or primitives, under this newly developed paradigm, are insufficient for the construction of secure IO schemes. In particular, we rule out constructions of IO from both various traditional long-established primitives and some of the more recent advanced objects. Furthermore, while this new extended framework facilitates a more comprehensive study of lower bounds for recent sophisticated primitives that are currently realized from more specialized cryptographic objects, it is of independent interest and opens up the opportunity to revisit and improve upon even the existing well-known impossibility results.

Acknowledgements

I would like to start by thanking my advisor Mohammad Mahmoody whom I've had the utmost pleasure and privilege of working with. His guidance and advice were invaluable in building and shaping my problem-solving skills. I am grateful for his willingness, responsiveness, and tolerance towards answering my (hopefully not too persistent) questions and satisfying my curiosities in this exciting eld of research.

I would like to thank my colleagues at the University of Virginia and at the University of California at Berkeley for indulging me with interesting and thought-provoking discussions on open problems and new insights including abhi shelat, Mohammad Hajiabadi, Saeed Mahloujifar, Soheil Nematihaji, Daniel Masny, Nico Doettling, and Peihan Miao. I would also like to give special thanks to Sanjam Garg who was kind enough to allow me to visit UC Berkeley. During my stay, I have bene ted greatly from Sanjam's meetings and discussions, and gained much from the captivating theory talks that were presented by other researchers and experts from within (and without) the department.

I would like to thank the committee members - David Evans, Jack Davidson, Denis Nekipelov, and Sanjam Garg - who contributed their time to be part of this work through their useful feedback and discussions.

I would also like to extend my thanks towards Kuwait University who were gracious enough to support and sponsor my graduate studies in the United States.

Last but not least, I would like to thank my family and friends back at home; your words of encouragement and overwhelming show of support have kept my spirits high and taught me to stay positive throughout this incredible and enlightening journey. This work is dedicated to you all.

Contents

List of Figures

Chapter 1

Introduction

Program obfuscation is the task of hiding any information embedded within any given program without a ecting the functionality of the program. Obfuscation was rst introduced by Hada in [Had00] and later formalized and studied in detail by Barak, Goldreich, Impagliazzo, Rudich, Sahai, Vadhan, and Yang in [BGI+01]. The strongest notion of obfuscation, called *virtual black-box* (VBB) obfuscation, has the property that an attacker with access to the obfuscated program learns no more information about the program than if it had just \black-box" access to this program - that is knowing only the input-output behaviour of the program. While this seems to be the ideal notion of security required by an obfuscation mechanism, it was shown in [BGI+01, GK05] that it is impossible to achieve in general. However, this does not preclude that certain restricted classes of functions can be obfuscated under this notion. Indeed, a series of works have demonstrated the possibility of VBB obfuscation for speci c families of functions, including point functions [CMR98, LPS04, Wee05, CD08, BC10], re-encryption functions [HRsV07], membership-checking in constant-dimension hyperplanes [CRV10], and conjunctions [BR13].

In order to circumvent this impossibility result on VBB obfuscation, [BGI+01] de ned an alternative weaker notion of obfuscation called *indistinguishability obfuscation* (IO). Informally, the security of IO states that an attacker cannot distinguish between obfuscations of two *functionally-equivalent* circuits of the same size. At rst glance, the security guarantee might seem too weak to enable its use in practice as there is no explicit assurance that information will remain hidden after obfuscation. Nevertheless, such a security notion was deemed to be equivalent to what [GR07] referred to as \best-possible" obfuscation where the obfuscated circuit leaks as much information as any other circuit of the same size and input-output behaviour.

It was not until the seminal work of Garg, Gentry, Halevi, Raykova, Sahai, and Waters [GGH+13b] that interest in IO was reinvigorated as the authors in that work provided the rst candidate construction of an IO scheme from multi-linear map assumptions [GGH13a, CLT13, GGH15], which were believed to be, at the time, hard to break. This started a series of works exploring the di erent possible applications of IO, some of which were not even known to be feasibly realizable from previous tools. Such applications include functional encryption [GGH+13b, Wat15], witness encryption [GGSW13], public-

key and deniable encryption [SW14], multi-party computation [GGHR14, GP15, CGP15], oblivious transfer [SW14], veri able searchable encryption [CYG+15], identity-based fully homomorphic encryption [CM14], and many more.

Following the initial candidate construction of IO [GGH+13b], a long line of work [AGIS14, MSW14, PST14, AB15, Zim15, BMSZ15, GMM+16, Lin16, LV16, Lin17] focused on building new and improved IO constructions that either use more robust assumptions or possess security properties that are proven to exists in some idealized model. Furthermore, all these constructions use multi-linear maps as their building block in some way or another. However, several attacks [CHL+15, HJ16, CGH+15, CLLT16, MSZ16] against multi-linear maps proved that these assumptions were not as sturdy as initially believed to be, and this created an uncertainty towards the security of the IO constructions that are based on these broken assumptions.

As a result of the above problem, it has become increasingly desirable to nd more standard, time-tested assumptions (e.g. Discrete Log, Decisional Di e-Hellman, etc.) on which the security of IO constructions can be based on, instead of the more exotic assumptions that are not well-understood. So far, the only known alternative constructions of IO go through what is known as \functional" encryption [AJ15, BV15], which can roughly be described as an encryption scheme with expressive access control mechanisms. However, all known functional encryption schemes that are strong enough to give IO are also based on multilinear maps. This leads us to the motivation of this work as we strive to investigate the reasoning behind why IO constructions gravitate towards using such strong assumptions.

1.1 Motivation

An important line of work in theoretical cryptography is the study of the hardness of the underlying assumptions on which computationally secure cryptographic tasks are built upon. By analyzing the \complexity" of computational assumptions, one can determine their relative strength and the resulting implications between them. More importantly, this allows us to argue whether certain \simpler" primitives and/or assumptions can be used to construct more sophisticated tools that have more useful, impactful applications. For example, it is well known that trapdoor permutations, a standard well-studied cryptographic tool, are su cient to be used as low-level building blocks for constructing public-key encryption schemes [Sah99, DDN00]. On the other hand, it was proven that the existence of just one-way functions, which while is a more desirable assumption, is simply not enough to build public-key encryption schemes (in a black-box way) [IR89]. The latter result is said to be a *lower bound* on the complexity of public-key encryption since we require an assumption that is at least stronger than just the existence of one-way functions in order to construct such a scheme, and the ultimate goal is to nd the best (i.e. weakest assumption) to realize any scheme.

The main goal of this thesis is to extend the above line of work on lower bounds but for the case of indistinguishability obfuscation. More speci cally, we would like to make progress towards answering the following question:

What assumptions could be used for realizing indistinguishability obfuscation?

As is the case for any other primitive, an important motivation behind determining the lower bounds of IO, which is our main focus in this work, is that it helps in finding weaker well-studied assumptions on which to base IO. This is more so the case for IO as it is a relative newcomer to the scene and researchers that study it are yet to be completely familiar with the complexity of such an object. Another motivation is that it guides research for building standard IO schemes and allows one to avoid using assumptions and/or tools that were proven to be insufficient on their own for achieving IO, instead focusing on alternative stronger, yet still robust, assumptions.

In order to even start looking at what assumptions or primitives are sufficient (or not) for building secure IO constructions, one needs to first define the framework under which such constructions exist. More generally, we need to ask what it means to have a secure construction of some primitive Q from another primitive P. In particular, one cannot hope to rule out *all* constructions of Q from P especially considering that one can (perhaps injudiciously) assumes that Q exists unconditionally. Therefore, in order to capture a meaningful construction of one primitive from another, [IR89, RTV04] proposed a taxonomy that classifies any given such construction as one of several types. Of particular interest are *fully black-box* constructions, which $(\mathbf{1})$ allow the construction of primitive Q to only use primitive P in a \black-box" way (i.e. knowing only the input-output behaviour of P and without knowledge of the actual implementation of P) and $(\mathbf{2})$ allow the security of Q to be based on the security of P. Most constructions in cryptography are black-box constructions and they are often desirable as they lead to more practical implementations. As such, ruling out such constructions of a primitive Q from another primitive P (which we call a *black-box separation* of Q from P) would prove that a large class of natural constructions of Q from P are indeed impossible.

On the other hand, one major drawback of ruling out just fully black-box constructions Q from P is that it does not necessarily mean that there are other *non-black-box* techniques that allow Q to be constructed from P. Consider for example a possible construction of some primitive Q from another primitive P where Q may need to feed the code of P back to P itself. Such a construction would not violate any black-box impossibility result that says Q cannot be based on P in a black-box way. In fact, it might very well be that such a candidate construction would circumvent this impossibility result.

The case with currently known IO constructions from functional encryption is that they all make non-black-box use of the underlying primitive. Consequently, in order to rule out any constructions of IO from any primitive P that has a similar flavor to functional encryption, it would not be very useful to consider ruling out only black-box constructions of IO from P since any such potential construction would most likely use P in a non-black-box way. These potential constructions do not fall under the classical framework of [RTV04] so we cannot rule them out using standard black-box separation techniques. As a result, this motivates us to extend the framework of [RTV04] to capture IO constructions of this form (and perhaps even other primitives of similar nature) in order to begin proving our impossibility results.

1.2 Contributions

In light of the motivation behind this thesis, we will highlight the three main contributions of this work.

1.2.1 Impossibility of VBB Obfuscation in Idealized Models

Given that the central focus of this work is to provide lower bounds on indistinguishability obfuscation, it is natural to ask whether we can also provide similar lower bounds for virtual black-box (VBB) obfuscation in idealized models where algorithms have access to oracles that are provably secure. While the impossibility of VBB for general circuits in the plain model (where no oracle exists) has already been established, we extend this negative result by proving that VBB is impossible to achieve *even* in certain more powerful/expressive idealized models. We also note that despite the impossibility of general plain-model, a construction for VBB obfuscation in some idealized model could still be used as a practical heuristic obfuscation mechanism once instantiated with a strong hash function (such as SHA-3). This would be in analogy with the way random oracle-based constructions of other primitives are widely used in practice *despite* the impossibility results of [CGH04].

We will use the work of Canetti, Kalai, and Paneth [CKP15] as our starting point for this part. It was shown there that a VBB obfuscator that has access to a random oracle I (an ideal random function) can be transformed in to an approximately correct VBB obfuscator in the plain model where no oracle I exists. However, assuming trapdoor permutations (which is a reasonable and well-believed assumption), the work of [BP13] have shown that even approximately correct VBB obfuscation does not exist in the plain model. Together with [CKP15], this rules out VBB obfuscation in the random oracle model.

Using the same reasoning for di erent idealized oracles which we list below, we show how to remove or \compile out" these oracles from the VBB obfuscator and get an approximately correct obfuscator in the plain model thus allowing us to argue the impossibility of such an obfuscation in these ideal models. The ideal models that we consider are the following (see Section 2.3 for formal de nitions):

Random Trapdoor Permutation Model

Generic Group Model (over any abelian group)

Generic $O(1)$-degree Graded Encoding Model (over any nite ring)

This complements the results of [BR14, BGK+14] where they show that general VBB obfuscation is indeed possible in the idealized graded encoding model of some polynomial degree. This contribution is discussed in Chapter 4 and appears in [MMN16a]. We also mention the concurrent work of [PS16], which prove that VBB obfuscation is impossible in the idealized constant-degree GEM over nite *elds*. Part of our work extends that result by also proving VBB obfuscation under idealized constant-degree GEM over nite *rings*.

1.2.2 Extending the Black-box Framework

The existing well-established framework of [IR89,RTV04] used for separating primitives from each other does not capture more sophisticated primitives whose constructions are naturally de ned in a way that permit some measure of \non-black-box" use of their underlying components (e.g. using the code or implementation of the underlying object). In order to even start working on separating IO from more powerful primitives, this framework needs to be extended in a way that re ects such constructions.

In more detail, let us rst consider what it means to have a construction of a primitive in the traditional black-box framework. Informally, we say that we have a *fully black-box construction* of a primitive Q from another primitive P if we can construct an algorithm Q that correctly implements Q using only the input-output behavior of any implementation P of P, and if P is secure then we can show that Q is secure as well.

In this work, we would like to consider constructions of primitives that do not fall under the aforementioned types of constructions. For example, a known construction of IO is based on a strong form of an encryption mechanism called functional encryption (FE) and uses the subroutines of FE in a non-black-box way. In particular, one of the subroutines of FE is a key generation algorithm and the construction of IO needs to supply the *code* of FE to the key generation algorithm. This is currently not captured by the black-box framework of [IR89,RTV04] since the correctness condition requires that Q must use P in a black-box way.

However, we observed that for these types of non-black-box use of P, Q needs only to \plant" oracle gates (i.e. subroutine calls) to P when calling P subroutines. Thus, our rst line of thought is to extend this de nition so that the construction allows Q to use P^P in a black-box way, where P^P allows Q to call P with circuits/Turing machines that have oracle gates to P. This almost gives us what we want, but we still need to consider various subtle issues regarding what constitutes as a valid extension of some primitive P. More speci cally, we require a de nition for *extended* primitives (i.e. P^P) that (1) covers all the known positive constructions of this form and (2) allows us to apply our separation techniques to rule out such extended primitives from constructing IO (or other similar primitives). We note here that this de nition generalizes previous models that were developed in [BKSY11,AS15] where P accepts only circuits with one-way function gates.

In Chapter 6 we present and de ne an extended black-box framework, called the *monolithic framework*, which allows us to model constructions that make use of a large and natural class of commonly-used non-black-box techniques in order to later prove that such techniques are not helpful for building IO from other strong primitives. This contribution appears in [GMM17a].

1.2.3 Separating IO from other Assumptions

As the main goal this work, we prove black-box separation results for IO from standard assumptions and, by leveraging the newly developed monolithic framework, from even more powerful primitives as well. The standard primitives that we rule out include one-way func-

tions, trapdoor permutations, and hash functions. Furthermore, we show that assumptions based on generic groups and generic encoding models (e.g. discrete log and \low-degree" multi-linear assumptions) are not su cient for building IO under the black-box framework. We also examine more sophisticated primitives such as predicate encryption [BSW11], fully homomorphic encryption [RAD78,Gen09], and witness encryption [GGSW13] and prove that such versatile tools are unable to build secure IO schemes using techniques captured by the monolithic framework.

In order to prove these separation results, we will leverage the result of [BBF16], where they show that statistically-secure approximate IO (that is correct on only some inputs) does not exist under reasonable complexity assumptions. Thus, by adopting the same approach that was used for compiling out idealized oracles in VBB obfuscators, we apply the technique here for any primitive P that we would like to separate IO from. In particular, we start with an idealized oracle I that securely instantiates P then we compile out I to get an approximately correct IO scheme for which we know there is an attack against by [BBF16].

This contribution is discussed in Chapters 5, 7, and 8, and it appears in [MMN+16b, GMM17a,GMM17b]. Figure 1.1 shows the known relationships surrounding IO and indicates our separation results. Note that the non-black-box constructions that we rule out (e.g. IO from fully homomorphic encryption) will be cast as *monolithic* constructions under our new framework.

1.3 Related Work

In this section we discuss the relevant related work regarding negative results on virtual black box obfuscation and indistinguishability obfuscation. We also provide an overview of previous results that prove non-black-box separations in various other areas in cryptography.

Negative Results for Obfuscation While the previous work of [BGI+01, GK05] have shown that general VBB obfuscation is impossible, this still left open the question of whether such a strong form of obfuscation can perhaps exist in an idealized world where powerful black-box algorithms (oracles) aid the obfuscation in ensuring the security of the scheme. The work of [BR14, BGK+14] prove that VBB obfuscation for general functions does indeed exist in what is known as the idealized poly-degree graded encoding model (GEM) where the obfuscation algorithm has access to an oracle that represents a multilinear map while capturing all the necessary properties required for security. This raises the question of whether other (perhaps weaker) idealized models are capable of allowing VBB obfuscation to exist. In [CKP15], it was shown that for the case that the idealized model is the random oracle model, VBB obfuscation for general circuits is impossible. Furthermore, in the concurrent work of [PS16], they prove that VBB obfuscation is impossible in the idealized poly-degree GEM over nite elds.

Turning to the known negative results for IO, the rst such result was shown by [GR07] where they proved that *statistically-secure*[1] IO is impossible under reasonable complexity

[1]While standard computational security of IO holds only against computationally bounded adversaries,

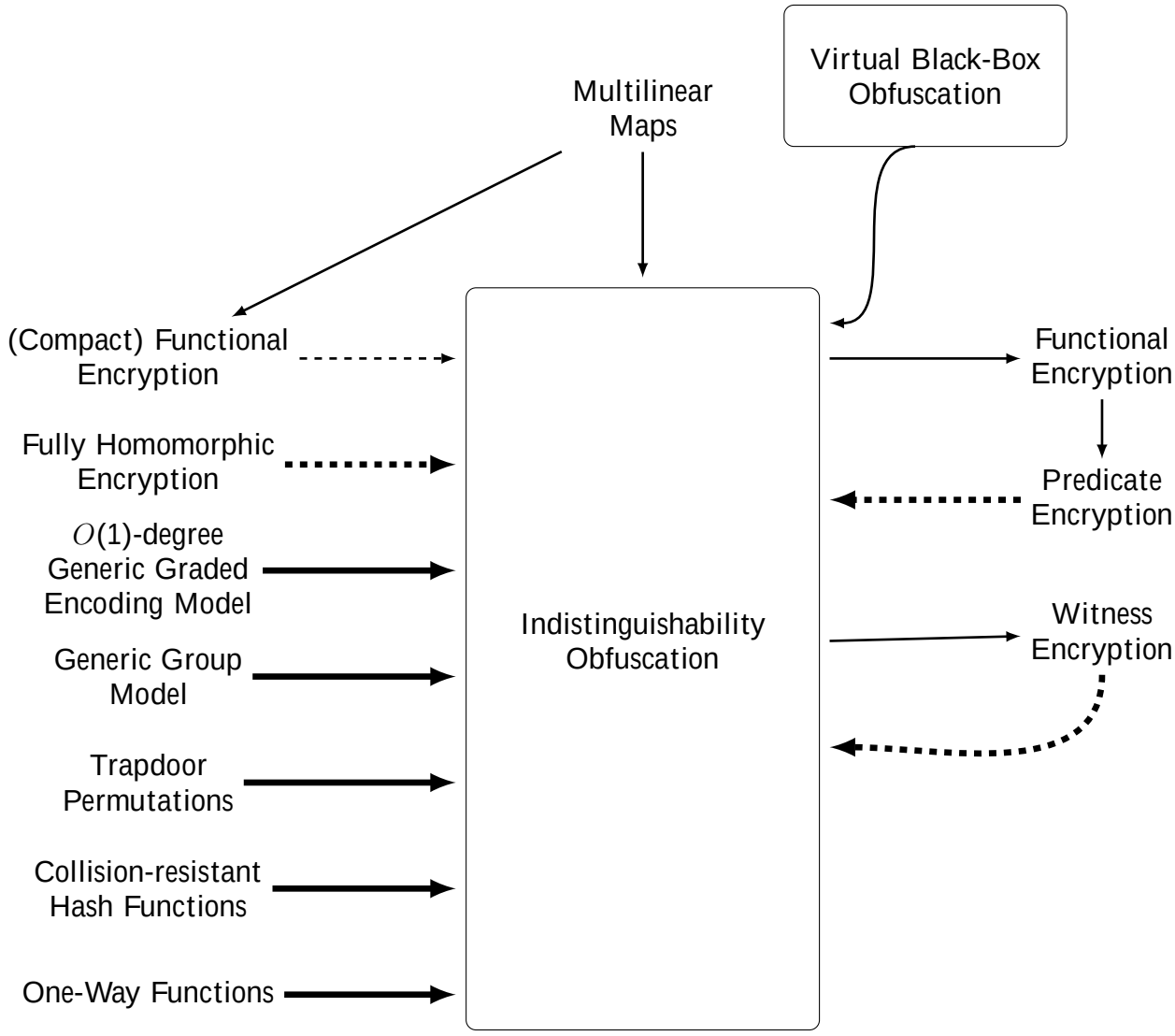

Figure 1.1: Relations between the assumptions surrounding indistinguishability obfuscation. Solid lines indicate black-box constructions while dashed lines indicates non-black-box constructions. The bold lines with on top indicate our contributions that prove the impossibility of basing IO on any of these underlying objects.

assumptions. More recently, [BBF16] extended the previous result by showing that even *approximately correct* statistically-secure IO is impossible (where approximate correctness implies that the obfuscated circuit may return an incorrect answer on some inputs). On a di erent but related note, the work of [AS15] provided the rst *limitation* of IO by proving that there exists no black-box construction of collision-resistant hash functions from an IO scheme that obfuscates circuits calling one-way functions[2].

Non-Black-Box Separations We mention here some of the previous work that prove the non-existence of non-black-box constructions from other primitives. These kinds of proofs are often extremely tied to the primitive whose construction is refuted and are thus somewhat di cult to generalize, but the consequences of the impossibility result are quite profound as they show that *even* non-black-box constructions of the primitive is impossible. For example, the work of [PTV11] showed that one can rule out a variety of non-black-box constructions of standard primitives (e.g. one-way permutations, collision-resistant hash functions) from one-way functions (under reasonable assumptions) as long as the security proof is black-box[3]. Furthermore, [GW11] show that succinct non-interactive argument systems cannot be based on many standard assumptions in a non-black-box way.

There also exists a few works in the literature that attempt to capture some non-black-box techniques within the black-box framework in order to rule out these constructions using standard techniques. We already mentioned that Asharov and Segev [AS15] have done exactly that by interpreting non-black-box constructions of hash functions from IO and one-way functions as black-box constructions of hash functions from IO *that obfuscate circuits calling one-way functions*. Similarly, the work of [BKSY11] interpreted non-black-box constructions of key-agreement protocols from zero-knowledge proofs and one-way functions as black-box constructions of key-agreement protocols from zero-knowledge proofs for statements that call one-way functions and then ruled out such constructions. We note that, while these works partially extended the notion of black-box constructions for the sake of ruling out a speci c primitive from a special class of primitives (those that, for example, take oracle circuits as input), we aim to generalize the black-box framework and establish theorems related to it so that it applies to any separations of the above form.

1.4 Organization

We start in Chapter 2 by presenting our notation that we will use in this work and list the de nitions of the cryptographic primitives relevant to our results. We present our results in two parts: the rst part discusses our black-box separations of IO while the second part is dedicated to our monolithic separations of IO from various advanced primitives. The rst part of our results starts with Chapter 3 where we review the classical black-box framework

statistically secure IO extends to computationally unbounded adversaries as well.

[2] This impossibility result is actually a proof for the non-existence of *partially* non-black-box constructions of hash functions from IO and one-way functions. Our work will generalize this notion in multiple directions.

[3] We discuss what it means to have a black-box security proof in Chapter 3.

due to [IR89, RTV04], and in Chapters 4 and 5 we present impossibility results for VBB obfuscation and IO from standard assumptions under the black-box framework. The second part of our results starts with Chapter 6 where we present our new monolithic framework. In Chapters 7 and 8 we will make use of this framework to prove even more separation results for IO constructions that are based on strong encryption primitives.

Chapter 2

Preliminaries

2.1 Notation

We let κ be the security parameter which, roughly speaking, determines the level of security of an instantiated cryptographic construction. We denote $f0,1g^n$ the set of n-bit strings and, for any string x, we let jxj be the bit-length of the string. We denote $f0,1g$ to be the set of all strings of arbitrary length. We use \$jj$" to concatenate strings and we use \,$" for attaching strings in a way that they could be retrieved. Namely, one can uniquely identify x and y from (x,y). For example (00jj11) = (0011), but $(0,011) \neq (001,1)$. We denote a vectors of dimension n as bold-faced variables $\mathbf{v} = (v_1,...,v_n)$.

Given any parameter n, we say that a function $p(n)$: N ! N is poly(n) if it is some polynomial in n. We say that a function $\epsilon(.)$ is a *negligible* function if for any polynomial $p(.)$, $\epsilon(n)$ $1/p(n)$ for su ciently large n.

Models of Computation. For any Turing machine (or algorithm) A, we say that A is uniform probabilistic polynomial time (PPT) or simply *e cient* if there exists a polynomial function $p(.)$: N ! N such that the running time of A on any input x is at most $p(jxj)$ (otherwise, we say it is ine cient). One can also consider the case that A is a *non-uniform* PPT Turing machine where, in addition to its input x, A would also accept an advice string $z_{jxj} \in f0,1g^{poly(jxj)}$ that depends on the length of the input. We call an algorithm $A^{(\cdot)}$ an e cient *oracle* (or *oracle-aided*) algorithm if it is e cient and is able to issue queries to some oracle O.

Definition 2.1.1 (Universal Circuit Evaluator)**.** For any family of circuits C with input size n, let $U(.,.)$ be a PPT oracle Turing machine that accepts as input a circuit $C \in C$ and an input $x \in f0,1g^n$. We call U a *universal circuit evaluator* if, for any (C,x), $U(C,x) = C(x)$.

Probability Distributions. For random variables X,Y, by X Y we denote the fact that X and Y are distributed identically. When writing the probabilities, by putting an algorithm A in the subscript of the probability (e.g., $Pr_A[\,]$) we mean the probability is over A's randomness. For any given probability distribution $\mathbf{D}$, we denote x $\mathbf{D}$ as sampling

from this distribution and obtaining a sample x from the support of $\mathbf{D}$. We denote (X, Y) to be the statistical distance between the two random variables. We say that X and Y are *statistically* indistinguishable (X, Y) negl(κ).

2.2 Measure-Theoretic Lemmas

By a *probability space* we mean a *measure space* with total measure equal to one, and by Pr[E] we denote the measure of the measurable set E. For a sequence of measurable sets $E = (E_1, E_2, \ldots)$ de ned over some measure space, the limit supremum of E is de ned as limSup(E) = $\bigcap_{n=1}^{1} \bigcup_{m=n}^{1} E_m$. It can be shown that limSup(E) is measurable if E_i is so for all i.

Lemma 2.2.1 (Borel{Cantelli [Bor09, Can17]). *Let $E = (E_1, E_2, \ldots)$ be a sequence of measurable sets over some probability space, and $\sum_{n=1}^{1}$ Pr[E_i] < 1. Then* limSup(E) *has measure zero.*

The following lemma follows from Exercise 2 of Section 7.3 of [GS01]. For completeness we give a proof using continuity of probability.

Lemma 2.2.2. *If $E = (E_1, E_2, \ldots)$ is a sequence of measurable sets over some probability space, and* Pr[E_i] δ *for all i 2* N*, then* Pr[limSup(E)] δ.

Proof. We use the following well-known lemma whose proof could be found in [Ale03] Proposition 37, Part (iii).

Lemma 2.2.3 (Continuity of Probability). *Let B_1 B_2 $\ldots$ be a sequence of measurable sets over some measure space, and* Pr[B_1] < 1. *Then* Pr $[\bigcap_{n=1}^{1} B_n]$ = lim$_{n \,!\, 1}$ Pr $[B_n]$.

Now let $B_n = \bigcup_{m=n}^{1} E_m$, and so limSup($E$) = $\bigcap_{n=1}^{1} B_n$. Since the measure space is a probability space, thus we have Pr[B_1] 1, and we can apply the above lemma to conclude that

$$\lim_{n \,!\, 1} \Pr[B_n] = \Pr\left[\bigcap_{n=1}^{1} B_n\right] = \Pr[\text{limSup}(E)].$$

Finally, because Pr[B_n] Pr[E_i] δ for every n, we get δ lim$_{n \,!\, 1}$ Pr[B_n] = Pr[limSup(E)]. □

2.3 Generic/Idealized Models

An idealized model I is a randomized oracle; examples relevant to our work include the random oracle, random trapdoor permutation oracle, generic group model, and the graded encoding model, all of which we de ne more formally below. A sample I I of the oracle's distribution can (usually) be represented as a sequence $(I_1, I_2, \ldots)$ where I is the part of I that is de ned for security parameter κ. The distribution over the in nite object

I I could naturally be de ned through nite distributions D_i over the nite space of I_i. Caratheodory's extension theorem shows that such nite probability distributions could always be extended consistently to a measure space over the full in nite space of I I (see Theorem 4.6 of [Hol15] for a proof).

Definition 2.3.1 (Random Oracle Model)**.** In the random oracle model, all parties have access to a randomized oracle f such that for each input x, the answer $f(x)$ is uniformly (and independently of the rest of the oracle) distributed over $f0,1g^{jxj}$.

Definition 2.3.2 (Random Trapdoor Permutation)**.** For any security parameter κ, a random *trapdoor permutation* (TDP) oracle T consists of three subroutines (G, F, F^{-1}) as follows:

> $G(\ )$ is a random permutation over $f0,1g$ mapping trapdoors sk to a public indexes pk.
>
> $F[pk](x)$: For any xed public index pk, $F[pk](\ )$ is a random permutation over $f0,1g$.
>
> $F^{-1}[sk](y)$: For any xed trapdoor sk such that $G(sk) = pk$, $F^{-1}[sk](\ )$ is the inverse permutation of $F[pk](\ )$, namely $F^{-1}[sk](F[pk](x)) = x$.

Definition 2.3.3 (Generic Group Model (GGM) [Sho97])**.** Let $(G, \ )$ be any group of size N and let S be any set of size at least N. The *generic* group oracle $I[G \to S]$ (or simply I) is as follows. At rst an injective random function $\sigma : G \to S$ is chosen, and two type of queries are answered as follows.

> **Labeling Queries.** Given $g \in G$ oracle returns $\sigma(g)$.
>
> **Addition Queries.** Given y_1, y_2, if there exists x_1, x_2 such that $\sigma(x_1) = y_1$ and $\sigma(x_2) = y_2$, it returns $\sigma(x_1 \ x_2)$. Otherwise it returns $?$.

Definition 2.3.4 (Degree-d Ideal Graded Encoding Model (GEM))**.** The oracle $M_R^d = (\text{enc}, \text{zero})$ is stateful and is parameterized by a ring R and a degree d and works in two phases. For each l the oracle $\text{enc}(\ , l)$ is a random injective function from the ring R to the set of labels S.

1. Initialization phase: In this phase the oracle answers $\text{enc}(v, l)$ queries and for each query it stores (v, l, h) in a list L_O.
2. Zero testing phase: Suppose $p(\ )$ is a polynomial whose coe cients are explicitly represented in R and its monomials are represented with labels $h_1, \ldots, h_m$ obtained through $\text{enc}(\ , \ )$ oracle in phase 1. Given any such query $p(\ )$ the oracle answers as follows:
   (a) If any h_i is not in L_O (i.e., it is not obtained in phase 1) return false.
   (b) If the degree of $p(\ )$ is more than d then return false.
   (c) Let $(v_i, l_i, h_i) \in L_O$. If $p(v_1, \ldots, v_m) = 0$ return true, otherwise false.

Definition 2.3.5 (Generic Algorithms). Let A^I be an algorithm (or a set of interactive algorithms $A = \{A_1, A_2, \ldots\}$) where I is the generic group oracle or the graded encoding oracle. We call A^I a *generic algorithm* in the generic group model (resp. graded encoding model) model if it never asks any addition (resp. zero-testing) query that is answered as $?$. Namely, only queries are asked for which the labels are previously obtained.

In this work we only use *sparse* encodings in which $|S|/|G| = \kappa^{-\omega(1)}$ (for the generic group model) or $|S|/|R| = \kappa^{-\omega(1)}$ (for the graded encoding model) where κ is the security parameter. Therefore, the execution of poly-time algorithms in this model will be statistically close to being generic.

2.4 Basic Primitives

In this section we list the definitions of various common and basic cryptographic primitives and assumptions which we will use or refer to throughout this work.

Definition 2.4.1 (One-way Function (OWF)). Let $f : \{0,1\}^* \to \{0,1\}^*$ be a function. We call f a *one-way function* if the following conditions are satisfied:

Completeness: For any input x, the running time of $f(x)$ is at most poly($|x|$).

One-way: For any PPT adversary A, there exists a negligible function negl such that for any security parameter κ:

$$\Pr_{x \xleftarrow{\$} \{0,1\}^\kappa}[A(f(x)) \in f^{-1}(f(x))] \leq \mathsf{negl}(\kappa)$$

Definition 2.4.2 (Collision-resistant Hash Functions (CRHF)). Let $h = \{h_i : \{0,1\}^n \to \{0,1\}^m\}_{i \in \mathbf{I}}$ be a family of functions where $m < n$ and $\mathbf{I}$ is efficiently sampleable. We call h a *collision-resistant hash function* (family) if the following conditions are satisfied:

Completeness: The running time of $h_i(x)$ for any i and x is at most poly($|x|$).

Collision-resistant: For any (even non-uniform) PPT adversary A, there exists a negligible function negl such that for any security parameter κ:

$$\Pr_{i \leftarrow \mathbf{I}}[(x, x') \leftarrow A(1^\kappa, i) : x' \neq x \wedge h_i(x) = h_i(x')] \leq \mathsf{negl}(\kappa)$$

2.5 Obfuscation

Below we give a direct formal definition for approximately correct virtual black-box (VBB) obfuscation in idealized models. The (standard) definition of VBB is equivalent to 0-approximate VBB in the plain model where no oracle is accessed.

Definition 2.5.1 (Virtual Black-box Obfuscation in Idealized Models)**.** For security parameter κ 2 N and function ϵ: N ! [0, 1], an ϵ-approximate virtual black-box (VBB) obfuscation scheme in an idealized model I consists of two oracle PPT algorithms Obf = (vO, Ev) de ned as follows:

Obfuscator vO is a PPT that takes as inputs a circuit C and a security parameter 1 and outputs a \circuit" B.

Evaluator Ev takes as input (B, x) and outputs y.

The completeness and security conditions assert that:

Correctness: For any circuit C of size n and input size m:

$$\Pr_{x \quad f0;1g^m}[\mathsf{Ev}^I(B, x) \neq C(x)] \quad \epsilon(\kappa)$$

where B vO^I(1 , C), and the probability is over the choice of input x, the oracle I, and the internal randomness of vO. If ϵ = 0, then it is a perfectly correct obfuscator.

Virtual Black-box Security: For every PPT adversary A, there exists a PPT simulator Sim and a negligible function μ such that for all n 2 N and circuits C 2 $f0, 1g^n$:

$$\left|\Pr[A^I(\mathsf{vO}^I(1 , C)) = 1] \quad \Pr[\mathsf{Sim}^C(1 , 1^{jCj}) = 1]\right| \quad \mu(\kappa)$$

where the probability is over I and the randomness of A, Sim, and vO.

Definition 2.5.2 (Indistinguishability Obfuscation (IO) [BGI+01])**.** An indistinguishability obfuscation (IO) scheme consists of two subroutines:

Obfuscator iO is a PPT that takes as inputs a circuit C and a security parameter 1 and outputs a \circuit" B.

Evaluator Ev takes as input (B, x) and outputs y.

The completeness and security conditions assert that:

Completeness: For every C, with probability 1 over the randomness of iO, we get B iO(1 , C) such that: For all x it holds that Ev$(B, x) = C(x)$.

Security: For every poly-sized distinguisher D there exists a negligible function $\mu(\ )$ such that for every two circuits C_0, C_1 that are of the same size and compute the same function, we have:

$$j \Pr_{\mathsf{iO}}[D(\mathsf{iO}(1 , C_0)) = 1] \quad \Pr_{\mathsf{iO}}[D(\mathsf{iO}(1 , C_1)) = 1]j \quad \mu(\kappa)$$

Definition 2.5.3 (Approximate IO)**.** For function 0 $< \epsilon(n)$ 1, an ϵ-approximate IO scheme is de ned similarly to an IO scheme with a relaxed completeness condition:

ϵ-approximate completeness. For every C and n we have:

$$\Pr_{x,\mathsf{iO}}[\mathsf{Ev}(B,x) = C(x) \mid B = \mathsf{iO}(1^\kappa, C)] \geq 1 - \epsilon(\kappa)$$

Definition 2.5.4 (Approximate Statistical Correlation IO [BBF16])**.** A PPT IO scheme $(\mathsf{iO}, \mathsf{Ev})$ is an (ϵ, δ)-*approximate statistical correlation IO* (CIO for short) if:

Approximate correctness: $\Pr[\mathsf{Ev}(B,x) \neq C(x)] \leq \epsilon(|C|)$ where $B = \mathsf{iO}(1^\kappa, C)$ and the probability is over the randomness of the obfuscator and the input x.

Statistical correlation: For every pair of circuits C_1, C_2 of the same size n, the statistical distance between $\mathsf{iO}(1^\kappa, C_1)$ and $\mathsf{iO}(1^\kappa, C_2)$ (both defined over the randomness of iO) is at most $\delta(n)$.

A computational variant of Definition 2.5.4 can be defined analogously:

Definition 2.5.5 (Approximate Computational Correlation IO)**.** A PPT IO scheme $(\mathsf{iO}, \mathsf{Ev})$ is an (ϵ, δ)-approximate *computational* CIO if it satisfies the same correctness condition as approximate statistical CIO and:

Computational correlation: For every poly-time adversary A and for every pair of circuits C_1, C_2 of equal size n, it holds that $\Pr[A(\mathsf{iO}(1^\kappa, C_1)) = 1] - \Pr[A(\mathsf{iO}(1^\kappa, C_2))] \leq \delta(n)$.

The Evaluation Algorithm. All of the definitions of the obfuscation scheme above have a subroutine for evaluating the obfuscated code. The reason for defining the evaluation as a subroutine of its own is that when we want to construct IO in oracle/idealized models, we allow the obfuscated circuit to call the oracle as well. Having an evaluator subroutine to run the obfuscated code allows to have such oracle calls in the framework of black-box constructions of [RTV04] where each primitive Q is simply a class of acceptable functions that we (hope to) efficiently implement given oracle access to functions that implement another primitive P.

However, for simplicity and whenever it is unambiguous, we can consider the Ev algorithm as running the obfuscated code B on input x and we can then simply write $\mathsf{iO}(C)(x)$ as being an equivalent notation to $\mathsf{Ev}(\mathsf{iO}(C), x)$.

2.6 Encryption Primitives

In this section we present the definitions for the different encryption primitives that we will deal in this work. We will frequently refer to them during our main results whenever it is necessary and relevant to the current section.

2.6.1 Witness Encryption

Definition 2.6.1 (Witness Encryption (WE) indexed by veri er V [GGSW13]). Let L be an **NP** language with a corresponding e cient relation veri er V (that takes instance a and witness w and either accepts or rejects). A *witness encryption* scheme for relation de ned by V consists of two PPT algorithms $(\mathsf{Enc}, \mathsf{Dec}_V)$ de ned as follows:

> $\mathsf{Enc}(1^\kappa, a, m)$: given an instance $a \in \{0,1\}^*$ and a message $m \in \{0,1\}^*$, and security parameter κ (and randomness as needed) it outputs $c \in \{0,1\}^*$.
>
> $\mathsf{Dec}_V(w, c)$: given ciphertext c and \witness" string w, it either outputs a message $m \in \{0,1\}^*$ or $?$.

We also need the following completeness and security properties:

> **Completeness:** For any security parameter κ, any (a, w) such that $V(a, w) = 1$, and any m it holds that
>
> $$\Pr_{\mathsf{Enc};\mathsf{Dec}_V}[\mathsf{Dec}_V(w, \mathsf{Enc}(1^\kappa, a, m)) = m] = 1$$
>
> **Security:** For any PPT adversary A, there exists a negligible function $\mu(.)$ such that for all $a \notin L_V$ (i.e., that there is no w for which $V(a, w) = 1$) and any $m_0 \neq m_1$ of the same length $|m_0| = |m_1|$ the following holds:
>
> $$|\Pr[A(\mathsf{Enc}(1^\kappa, a, m_0)) = 1] - \Pr[A(\mathsf{Enc}(1^\kappa, a, m_1)) = 1]| \leq \mu(\kappa)$$

When we talk about the witness encryption as a primitive (not an indexed family) we refer to the special case of the `complete' veri er V which is a universal circuit algorithm and $V(w, a) = 1$ if $a(w) = 1$ where a is a circuit evaluated on witness w.

We also de ne variants of witness encryption, which are both a strengthening of the De nition 2.6.1, one of them strengthens WE's functionality and the other one strengthens its security. Therefore these variants of WE are incompatible in their capabilities and, hence, a witness encryption scheme may possess either one or the other extra feature (but not both).

Definition 2.6.2 (Instance-revealing Witness Encryption (IRWE)). A witness encryption scheme is said to be *instance-revealing* if it satis es the properties of De nition 2.6.1 and, in addition, includes the following subroutine.

> **Instance-Revealing Functionality**: $\mathsf{Rev}(c)$ given ciphertext c outputs $a \in \{0,1\}^s \cup \{?\}$, and for every a, m, κ:
>
> $$\Pr_{\mathsf{Enc};\mathsf{Rev}}[\mathsf{Rev}(\mathsf{Enc}(1^\kappa, a, m)) = a] = 1.$$

Definition 2.6.3 (Instance-hiding Witness Encryption (IHWE)). A witness encryption scheme is said to be *instance-hiding* if it satis es the properties of De nition 2.6.1 except that the security property is replaced with the following (stronger) security guarantee:

Instance-Hiding Security: For any PPT adversary A, there exists a negligible function $\mu(.)$ such that for all a_0, a_1 2 V for which $ja_0j = ja_1j$ and m_0, m_1 for which $jm_0j = jm_1j$, the following holds:

$$|\Pr[A(\text{Enc}(1^\kappa, a_0, m_0)) = 1] - \Pr[A(\text{Enc}(1^\kappa, a_1, m_1)) = 1]| \leq \mu(\kappa)$$

2.6.2 Predicate Encryption

Definition 2.6.4 (Predicate Encryption (PE) [BSW11]). Let $P_{K;A} : K \times A \to \{0,1\}$ be an e ciently computable predicate de ned over some key space K (that contains a special empty key ϵ) and attribute space A and P is an e cient Turing machine computing the relation $P_{K;A}$. A *predicate encryption* scheme for a class of predicates $P_{K;A}$ consists of four PPT algorithms (Setup, KGen, Enc, Dec_P) de ned as follows:

Setup(1^κ): given the security parameter, it outputs a master public key MPK and a master secret key MSK.

KGen(MSK, k): given $k \in K$ and the master secret key MSK $\in \{0,1\}^n$, outputs the decryption key sk_k. If $k = \epsilon$, it outputs ϵ.

Enc(MPK, (m, a)): given the master public key MPK, attribute $a \in A$, and message m, outputs ciphertext c.

$\text{Dec}_P(sk_k, c)$: given a secret key for $k \in K$ and a ciphertext c, outputs a string m (or ?).

The following completeness and security properties must be satis ed:

Completeness: For any security parameter κ, key $k \in K$, attribute $a \in A$, the following holds:

$$\text{Dec}_P(sk_k, \text{Enc}(\text{MPK}, (m, a))) = \begin{cases} (|a|, |m|) \text{ if } k = \epsilon \\ m \text{ if } k \neq \epsilon \text{ and } P(k, a) = 1 \\ ? \text{ Otherwise} \end{cases}$$

where $sk_k \leftarrow \text{KGen}(\text{MSK}, k)$ and $(\text{MSK}, \text{MPK}) \leftarrow \text{Setup}(1^\kappa)$.

Security: For any PPT adversary A, there exists a negligible function negl(.) such that the following holds:

$$\Pr[\text{IND}_A^{\text{PE}}(1^\kappa) = 1] \leq \frac{1}{2} + \text{negl}(\kappa)$$

where IND_A^{PE} is shown in Figure 2.1, and for each key query k that A sends to the KGen oracle, it must hold that $P(k, a_0) = P(k, a_1) = 0$.

Experiment $\mathsf{IND}_A^{\mathsf{PE}}(1^\kappa)$:

1. $(\mathsf{MSK}, \mathsf{MPK}) \leftarrow \mathsf{Setup}(1^\kappa)$
2. $(x_0, x_1) \leftarrow A^{\mathsf{KGen}(\mathsf{MSK},\cdot)}(\mathsf{MPK})$ where $x_b = (a_b, m_b)$ for $b \in \{0,1\}$, $|a_0| = |a_1|$ and $|m_0| = |m_1|$ and for each prior query k we require that $\mathsf{P}(k, a_0) = \mathsf{P}(k, a_1) = 0$
3. $b \xleftarrow{\$} \{0,1\}$
4. $c \leftarrow \mathsf{Enc}(\mathsf{MPK}, x_b)$
5. $b' \leftarrow A^{\mathsf{KGen}(\mathsf{MSK},\cdot)}(\mathsf{MPK}, c)$ where for each query k we require that $\mathsf{P}(k, a_0) = \mathsf{P}(k, a_1) = 0$
6. Output 1 if $b = b'$ and 0 otherwise.

Figure 2.1: The $\mathsf{IND}_A^{\mathsf{PE}}$ experiment used in defining the security for predicate encryption schemes

When P is clear from the context, we might simply write Dec instead of Dec_P.

We also present the definition of attribute-based encryption, which is a special case of predicate encryption where the attribute of the encrypted message could be potentially extracted efficiently from the ciphertext[1].

Definition 2.6.5 (Attribute-based Encryption (ABE))**.** An *attribute-based encryption* scheme is a predicate encryption scheme for a class of predicates $P_{K;A}$ (defined through efficient test $\mathsf{P}(k, a) \in \{0,1\}$) satisfying the properties of Definition 2.6.4 except that the completeness and security conditions are replaced with the following:

> **Completeness:** For any security parameter κ, key $k \in K$, attribute $a \in A$, and message $m \in M$, the following holds:
>
> $$\mathsf{Dec}_\mathsf{P}(sk_k, \mathsf{Enc}(\mathsf{MPK}, (m, a))) = \begin{cases} (a, |m|) \text{ if } k = \epsilon \\ m \text{ if } k \neq \epsilon \text{ and } \mathsf{P}(k, a) = 1 \\ \perp \text{ Otherwise} \end{cases}$$
>
> where $sk_k \leftarrow \mathsf{KGen}(\mathsf{MSK}, k)$ and $(\mathsf{MSK}, \mathsf{MPK}) \leftarrow \mathsf{Setup}(1^\kappa)$.
>
> **Security:** For any PPT adversary A, there exists a negligible function negl(.) such that the following holds:
>
> $$\Pr[\mathsf{IND}_A^{\mathsf{ABE}}(1^\kappa) = 1] \leq \frac{1}{2} + \mathsf{negl}(\kappa)$$
>
> where $\mathsf{IND}_A^{\mathsf{ABE}}$ is shown in Figure 2.2, and for each key query k that A sends to the KGen oracle, it must hold that $P(k, a) = 0$.

[1] The work of [BSW11] refers to this sub-class as predicate encryption with public index

Experiment $\mathsf{IND}_A^{\mathsf{ABE}}(1\ )$:
1. (MSK, MPK) Setup(1))
2. (x_0, x_1) $A^{\mathsf{KGen}(\mathsf{MSK},:)}$(MPK) where x_b = (a, m_b) for b 2 $f0, 1g$, jm_0j = jm_1j and for each prior query k we require that $P(k, a)$ = 0
3. b $\overset{\$}{}$ $f0, 1g$
4. c Enc(MPK, x_b)
5. b^0 $A^{\mathsf{KGen}(\mathsf{MSK},:)}$(MPK, c) where for each query k we require that $P(k, a)$ = 0
6. Output 1 if b = b^0 and 0 otherwise.

Figure 2.2: The $\mathsf{IND}_A^{\mathsf{ABE}}$ experiment used in de ning the security for attribute-based encryption schemes

2.6.3 Homomorphic Encryption

Definition 2.6.6 (Fully Homomorphic Encryption (FHE))**.** Let F be a PPT algorithm that accepts as input string f and messages $m_1, ..., m_t$ and outputs a string m^0. For any security parameter κ, a *homomorphic scheme* HE for F is composed of four PPT algorithms (Setup, Enc, Dec, Eval_F) de ned as follows:

Setup(1): given the security parameter κ, outputs the master public key MPK and the master secret key MSK.

Enc(MPK, m): given MPK and a message m 2 $f0, 1g$, outputs an encryption c.

Eval_F(MPK, $f, c_1, ..., c_t$): given master public key MPK, string f, and a sequence of ciphertexts c_1 = Enc(MPK, m_1), ..., c_t = Enc(MPK, m_t), outputs another ciphertext c_f.

Dec(MSK, c): given MSK and ciphertext c, outputs a bit m 2 $f0, 1g$.

A HE scheme is said to be *fully homomorphic* if the class of circuits supported for evaluation consists of all polynomially-sized circuits (i.e. $\mathbf{P/poly}$). Furthermore, an FHE scheme must satisfy the following properties:

Completeness: For any security parameter κ, string f and messages $m_1, \ldots, m_t$ 2 $f0, 1g$, it holds that:

$$\Pr[\mathsf{Dec}(\mathsf{MSK}, \mathsf{Eval}_\mathsf{F}(\mathsf{MPK}, f, c_1, ..., c_t)) = \mathsf{F}(f, m_1, ..., m_t)] = 1$$

where (MSK, MPK) Setup(1) and c_i = Enc(MPK, m_i) for i 2 $[t]$.

Compactness: There exists a xed polynomial $p(.)$ such that, for every string f, the size of the evaluated ciphertexts $j\mathsf{Eval}_\mathsf{F}(\mathsf{MPK}, f, c_1, \ldots, c_t)j$ is at most $p(\kappa)$ where, for all i 2 $[t]$, c_i Enc(MPK, m_i) for some m_i 2 $f0, 1g$.

Security: For any PPT adversary A, there exists a negligible function $\mu(.)$ such that the following holds:

$$|\Pr[A(\text{Enc}(\text{MPK}, 0)) = 1] - \Pr[A(\text{Enc}(\text{MPK}, 1)) = 1]| \leq \mu(\kappa)$$

where $(\text{MSK}, \text{MPK}) \leftarrow \text{Setup}(1^\kappa)$ and the probability is over the randomness of A and Enc.

We also provide two stronger variants of FHE where one may evaluate on multiple encryptions that were generated from different public keys but can only decrypt the newly evaluated ciphertext if one has access to some subset of the corresponding secret keys.

Definition 2.6.7 (Multi-key FHE (MFHE) [LTV12, MW16])**.** A multi-key FHE scheme is an FHE scheme where the evaluation sub-routine is replaced with the following:

$\text{Eval}(\vec{\text{MPK}}, f, c_1, ..., c_t)$: given a sequence of L distinct and independently generated keys $\vec{\text{MPK}} = (\text{MPK}_1, ..., \text{MPK}_L)$, a circuit $f \in \{0,1\}^*$, and a sequence of ciphertexts $(c_1, ..., c_t)$ where for all $i \in [t]$ there exists some $j \in [L]$ such that $c_i = \text{Enc}(\text{MPK}_j, m_i)$, outputs another ciphertext c_f.

Furthermore, the following properties must be satisfied:

Completeness: For any security parameter κ, messages $m_i \in \{0,1\}^*$ and function f, it holds that:

$$\Pr[\text{Dec}(\vec{\text{MSK}}, \text{Eval}(\vec{\text{MPK}}, f, c_1, ..., c_t)) = f(m_1, ..., m_t)] = 1$$

where $\vec{\text{MSK}} = (\text{MSK}_1, ..., \text{MSK}_L)$, $\vec{\text{MPK}} = (\text{MPK}_1, ..., \text{MPK}_L)$, and for all $j \in [L]$ we have $(\text{MSK}_j, \text{MPK}_j) \leftarrow \text{Setup}(1^\kappa)$ and for all $i \in [t]$ there exists $j \in [L]$ such that $c_i \leftarrow \text{Enc}(\text{MPK}_j, m_i)$.

Compactness: There exists a fixed polynomial $p(.,.)$ such that the size of the evaluated ciphertexts $|\text{Eval}(\vec{\text{MPK}}, f, c_1, ..., c_t)|$ is at most $p(\kappa, L)$ for any $f \in \{0,1\}^*$.

Security: For any PPT adversary A, there exists a negligible function $\mu(.)$ such that the following holds:

$$|\Pr[A(\text{Enc}(\text{MPK}, 0)) = 1] - \Pr[A(\text{Enc}(\text{MPK}, 1)) = 1]| \leq \mu(\kappa)$$

where $(\text{MSK}, \text{MPK}) \leftarrow \text{Setup}(1^\kappa)$ and the probability is over the randomness of A and Enc.

Definition 2.6.8 (Spooky Encryption [DHRW16])**.** Let F be a PPT algorithm that accepts as input a string f and messages $m_1, ..., m_t$ and outputs a sequence of messages $m'_1, ..., m'_t$. A F-*spooky encryption* scheme is an HE scheme for (possibly randomized) F where evaluation is replaced with the following:

$\mathsf{Eval}_F(f, (\mathsf{MPK}_1, c_1), ..., (\mathsf{MPK}_t, c_t))$: given a sequence of pairs of public key-ciphertext pairs (MPK_i, c_i), would output a sequence of ciphertexts $(c'_1, ..., c'_t)$.

Furthermore, the following properties must be satisfied:

Completeness: For any security parameter κ, string f and messages $(m_1, ..., m_t)$ where $m_i \in \{0,1\}^{\mathsf{poly}(\kappa)}$ for all $i \in [t]$, it holds that:

$$\{\mathsf{Dec}(\mathsf{MSK}_1, c'_1), ..., \mathsf{Dec}(\mathsf{MSK}_t, c'_t))\} \equiv_c \{F(f, m_1, ..., m_t)\}$$

where for all $i \in [t]$, $(\mathsf{MSK}_i, \mathsf{MPK}_i) \leftarrow \mathsf{Setup}(1^\kappa)$, $c_i \leftarrow \mathsf{Enc}(\mathsf{MPK}, m_i)$, and the evaluated ciphertexts are given as $c'_i \leftarrow \mathsf{Eval}_F(f, (\mathsf{MPK}_1, c_1), ..., (\mathsf{MPK}_t, c_t))$.

Security: For any PPT adversary A, there exists a negligible function $\mu(.)$ such that the following holds:

$$|\Pr[A(\mathsf{Enc}(\mathsf{MPK}, 0)) = 1] - \Pr[A(\mathsf{Enc}(\mathsf{MPK}, 1)) = 1]| \leq \mu(\kappa)$$

where $(\mathsf{MSK}, \mathsf{MPK}) \leftarrow \mathsf{Setup}(1^\kappa)$ and the probability is over the randomness of A and Enc.

In [DHRW16], a special case of spooky encryption called additive function sharing (AFS) spooky encryption was defined, which is essentially F-spooky encryption where F accepts and executes a circuit f_h that outputs a random n-out-of-n additive secret share of $h(m_1, ..., m_t)$. Note that F-spooky encryption supporting the class of all polynomially-sized circuits is a generalization of multi-key FHE [LTV12, MW16]. In particular, if we allowed repetition of public keys in the evaluation algorithm of the MFHE scheme, then we can define the evaluation algorithm of MFHE to simply use the evaluation of the spooky encryption scheme and output $c_f = (c'_1, ..., c'_t)$.

2.6.4 Functional Encryption

We will mainly be concerned with single-key functional encryption schemes which we define below so in the rest of this work whenever we refer to functional encryption, it is of the single-key type. We define a single-key functional encryption for function family $F = \{F_n\}_{n \in \mathbb{N}}$ (represented as a circuit family) as follows:

Definition 2.6.9 (Single-Key Functional Encryption [BV15])**.** A *single-key functional encryption* (FE) for function family F consists of three PPT algorithms (Setup, Enc, Dec) defined as follows:

Setup(1^κ): Given as input the security parameter 1^κ, it outputs a master public key and master secret key pair (MPK, MSK).

KGen(MSK, f): Given master secret key MSK and function $f \in F$, outputs a decryption key SK_f.

Enc(MPK, x): Given the master public key MPK and message x, outputs ciphertext $c \in \{0,1\}^p$.

Dec(SK_f, c): Given a secret key SK_f and a ciphertext $c \in \{0,1\}^m$, outputs a string $y \in \{0,1\}^s$.

The following completeness and security properties must be satisfied:

Completeness: For any security parameter κ, any $f \in \mathsf{F}$ with domain $\{0,1\}^n$ and message $x \in \{0,1\}^n$, the following holds:

$$\mathsf{Dec}(\mathsf{SK}_f, \mathsf{Enc}(\mathsf{MPK}, x)) = f(x)$$

where $(\mathsf{MPK}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\kappa)$ and $\mathsf{SK}_f \leftarrow \mathsf{KGen}(\mathsf{MSK}, f)$

Security: For any PPT adversary A, there exists a negligible function negl( ) such that:

$$\Pr[\mathsf{IND}_A^{1\mathsf{FE}}(1^\kappa) = 1] \leq \frac{1}{2} + \mathsf{negl}(\kappa),$$

where $\mathsf{IND}_A^{1\mathsf{FE}}$ is the following experiment.

Experiment $\mathsf{IND}_A^{1\mathsf{FE}}(1^\kappa)$:
1. $(\mathsf{MSK}, \mathsf{MPK}) \leftarrow \mathsf{Setup}(1^\kappa)$
2. $(f, x_0, x_1) \leftarrow A(\mathsf{MPK})$ where $|x_0| = |x_1|$ and $f(x_0) = f(x_1)$
3. $b \xleftarrow{\$} \{0,1\}, c \leftarrow \mathsf{Enc}(\mathsf{MPK}, x_b), \mathsf{SK}_f \leftarrow \mathsf{KGen}(\mathsf{MSK}, f)$
4. $b' \leftarrow A(\mathsf{MPK}, \mathsf{SK}_f, c)$
5. Output 1 if $b = b'$ and 0 otherwise.

Figure 2.3: The $\mathsf{IND}_A^{1\mathsf{FE}}$ experiment used in defining the security for functional encryption schemes

Efficiency: We define two notions of efficiency for single-key FE supporting the function family F:

- **Compactness:** An FE scheme is said to be *compact* if the size of the encryption circuit is bounded by some fixed polynomial $\mathsf{poly}(n, \kappa)$ where n is the size of the message, independent of the function f chosen by the adversary.[2]
- **Function Output Length:** An FE scheme is said to be t-*bit-output* if $\mathsf{outlen}(f) \leq t(n, \kappa)$ for any $f \in \mathsf{F}$, where $\mathsf{outlen}(f)$ denotes the output length of f. Given ciphertext length $p(n, \kappa)$, we say an FE scheme is *long-output* if it is $(p + i)$-bit-output for some $i \geq 1$ and *short-output* if it is only $(p - \omega(n + \kappa))$-bit-output where n is the size of the message.

Definition 2.6.10 (Functional Witness Encryption (FWE) [BCP14])**.** Let V be a PPT algorithm that takes as input an instance-message pair $x = (a, m)$ and witness w then outputs a bit. Furthermore, let F be a PPT Turing machine that accepts as input a witness w and a message m then outputs a string $y \in \{0,1\}^s$. For any given security parameter κ, a *functional witness encryption* scheme indexed with V and F consists of two PPT algorithms $P = (\mathsf{Enc}, \mathsf{Dec}_{\mathsf{V},\mathsf{F}})$ defined as follows:

[2] A couple of other weaker notions of compactness for FE have also been considered in the literature. However, all these notions are known to be "monolithically" equivalent to compact single-key FE. Therefore, we restrict our discussion just to compact single-key FE.

Enc(1 $^{\kappa}$, a, m) : given an instance $a \in \{0,1\}^*$, message $m \in \{0,1\}^*$, and security parameter κ, outputs $c \in \{0,1\}^*$.

$\mathsf{Dec}_{V;F}(w, c)$: given ciphertext c and "witness" string $w \in \{0,1\}^*$, outputs a message $m' \in \{0,1\}^*$.

A functional witness encryption scheme satisfies the following completeness and security properties:

Correctness: For any security parameter κ, any $m \in \{0,1\}^*$, and any $(w, (a, m))$ such that $V^P(w, a) = 1$, it holds that

$$\Pr_{\mathsf{Enc};\mathsf{Dec}}[\mathsf{Dec}_{V;F}(w, \mathsf{Enc}(1^{\kappa}, a, m)) = F^P(w, m)] = 1$$

Extractability: For any PPT adversary A and polynomial $p_1(.)$, there exists a PPT extractor E and a polynomial $p_2(.)$ such that for any security parameter κ, any a for which $V^P(w, a) = 1$ for some w, and any m_0, m_1 where $|m_0| = |m_1|$, if:

$$\Pr\left[A(1^{\kappa}, c) = b \mid b \xleftarrow{\$} \{0,1\}, c \leftarrow \mathsf{Enc}(1^{\kappa}, a, m_b)\right] \leq \frac{1}{2} + p_1(\kappa)$$

Then:

$$\Pr\left[\, E^A(1^{\kappa}, a, m_0, m_1) = w : V^P(w, a) = 1 \wedge F^P(w, m_0) \neq F^P(w, m_1) \,\right] \leq p_2(\kappa)$$

2.6.5 Universal Variants of Primitives

In all the primitives that we have defined in this section, there exists an efficiently computable function R that is part of the definition of this (family of) primitives. However, in this work, whenever we reference such a primitive, we will use a standard universal circuit evaluator function R_U (see Definition 2.1.1) that captures the most `complete' version of such a primitive and allows us to obtain the primitive in terms of any other relation. More specifically, the universal function is defined as follows for each relevant primitive:

Witness Encryption: The function (which is a relation here) R_U with corresponding verifier V is defined so that for any input pair (w, a), $V(w, a) = a(w)$. Thus, this relation gives rise to the language $L_V = \{a \mid \exists w : a(w) = 1\}$, which is essentially the language of circuit satisfiability.

Predicate and Attribute-based Encryption: The universal predicate class $P_{K;A}$ with corresponding P is defined so that for any $k \in K$ and $a \in A$, $P(k, a) = k(a)$. Thus, this relation gives rise to the language $L_P = \{k \mid \exists a : k(a) = 1\}$.

(Multi-key) FHE and Spooky Encryption: The algorithm F representing the scheme is defined so that for any input sequence $(f, m_1, ..., m_t)$, $F(f, m_1, ..., m_t) = f(m_1, ..., m_t)$.

Given a primitive defined with universal relation R_U with associated verifier V, we can construct the same primitive for any other arbitrary efficiently computable relation R' with

associated verifier V'. This is achieved by defining an instance a to be verified by V as a circuit $a := V'(., a')$ where a' is an instance to be verified by V'. In that case, we have that, for any (w, a'), $V'(w, a') = 1$ if and only if $V(w, a) = 1$ since $V(w, a) = a(w) = V'(w, a')$.

We can also define a symmetric variant of the universal relation. For example, for PE or ABE, we can let $P(k, a) = a(k)$ instead[3]. In either case, we will explicitly mention which input is to act as the circuit when we make use of these relations.

[3] In fact, these symmetric notions have been previously defined in the literature in the context of predicate encryption as key policy [GPSW06] and ciphertext policy [BSW07].

Part I

Black-box Separations for Indistinguishability Obfuscation

Chapter 3

The Black-box Framework

In this section, we present a background on the standard techniques used to prove black-box separation results. We first start by defining the various notions of black-box constructions then present how these constructions work in idealized models where access to randomized oracles is provided to all algorithms.

3.1 Black-box Constructions

Cryptographic *primitives* or objects, such as one-way functions and private-key encryption schemes, are traditionally defined as satisfying a *correctness* condition and a *security* condition. As such, any given *construction* of a primitive Q from another primitive P should satisfy both conditions in order to be a valid instantiation of Q. We refer to any construction of Q from P as being *black-box* if such a construction makes use of the input-output behaviour of P and not the actual implementation.

Impagliazzo and Rudich [IR89] were the first to formally study the power of black-box constructions that relativize to any oracle. Their notion was further explored in detail by Reingold, Trevisan, and Vadhan [RTV04]. The work of Baecher, Brzuska, and Fischlin [BBF13] further studied the black-box framework and presented variants of the definition of black-box constructions. We first start by recalling the definition of cryptographic primitives, and then will go over the standard notion of (fully) black-box constructions.

Definition 3.1.1 (Cryptographic Primitives [RTV04])**.** A *primitive* $P = (F, R)$ is defined as set of functions F and a relation R between functions. A (possibly inefficient) function $F : \{0,1\}^* \to \{0,1\}^*$ is a correct implementation of P if $F \in F$, and a (possibly inefficient) adversary A breaks an implementation $F \in F$ if $(A, F) \in R$.

Definition 3.1.2 (Black-box Constructions [RTV04])**.** A *black-box* construction of a primitive Q from a primitive P consists of two PPT algorithms (Q, S) defined as follows:

1. **Implementation**: For any oracle P that implements P, Q^P implements Q.

2. **Security reduction**: for any oracle P implementing $\mathcal{P}$ and for any (computationally unbounded) oracle adversary A breaking the security of Q^P, it holds that $S^{P,A}$ breaks the security of P.

By considering the role of the security parameter we can distinguish between attacks that succeed over an \in nite" vs. \all but nitely many" security parameters. Therefore, we will work with a more re ned de nition of cryptographic primitives (and constructions) where the parties also receive a security parameter κ as input. Thus, any function P implementing a primitive $\mathcal{P}$ will take as input a \security parameter" κ and, according to the standard de nition of security in cryptography, any adversary A who successfully breaks P would have to \win" over an *in nite* number of security parameters for a \noticeable" advantage (e.g. only winning over security parameters that are, say, powers of two will su ce to call it a successful attack).

Primitives with stronger hardness. The above de nition is for polynomially secure primitives. When the used primitive $\mathcal{P}$ is s-secure for a more quantitative bound $s(n)$ poly(n), the security reduction S could potentially run in longer running time as well so long as it holds that: when P, A are polynomial time, the total running time of the composed algorithm $S^{P,A}$ is also small enough to be considered a legitimate attack against the implementation P of $\mathcal{P}$.

Black-box Separations. In order to rule out black-box constructions we can use one of many various techniques that were developed in [IR89, GMR01, HR04], each of which have their advantages and di culties when ruling out certain classes of constructions. In most cases, one can easily see that in order to rule out a black-box construction of a primitive $\mathcal{Q}$ from another primitive $\mathcal{P}$, it su ces to show that, for any candidate construction Q of $\mathcal{Q}$, there exists an oracle O such that Q^O can be broken by some poly-query adversary A^O but any construction P of $\mathcal{P}$ is secure against any adversary that tries to break it.

3.1.1 Variants on Black-box Constructions

In the following more relaxed form of constructions, the security reduction can depend arbitrarily on the adversary but it still treats the implementation of the used primitive in a black-box way.

Definition 3.1.3 (Semi-black-box constructions [RTV04])**.** A *semi-black-box* construction of a primitive $\mathcal{Q}$ from a primitive $\mathcal{P}$ is de ned similarly to the fully black-box De nition of 3.1.2 with the following di erence in the security reduction:

> For any oracle P implementing $\mathcal{P}$ and any *e cient* oracle-aided adversary A^P who breaks the security of Q^P it holds that $S^P(A)$ breaks the security of P. Note that since A's description is e cient it could indeed be given to S in a non-black-box way.

Efficiency of adversary. We used the term *e cient* in an unspeci ed way so that it could be applied to complexity classes beyond polynomial time. For example, using a quasi-polynomially secure primitive *P* to construct a polynomially secure primitive *Q* would require a security primitive that is more relaxed and could lead to a quasi-polynomial (as opposed to polynomial) time attack against P using any polynomial-time attacker against Q^P.

3.1.2 Black-box Constructions in Idealized Models

Definition 3.1.4 (Oracle- xed constructions in idealized models)**.** We say a primitive *P* exists relative to the *randomized* oracle (or idealized model) *I* if there is an oracle-aided algorithm P such that:

1. **Completeness:** For every instantiation I *I*, it holds that P^I implements *P* correctly.
2. **Security:** Let A be an oracle-aided adversary A^I where the complexity of A is bounded by the speci ed complexity of the attacks for primitive *P*. For example if *P* is polynomially secure (resp., quasi-polynomially secure), then A runs in in polynomial time (resp., quasi-polynomial time). For every such oracle-aided A, with measure one over the sampling of the idealized oracle $I \xleftarrow{\$} I$, it holds that A does *not* break the security of P^I.

We call P a *black-box* construction of *P* relative to *I* if the security property holds also in a \black-box" way de ned as follows:

> **Black-box security:** Let A be an oracle-aided adversary A^I where the *query complexity* of A is bounded by the speci ed complexity of the attacks for primitive *P*. For example if *P* is polynomially secure (resp., quasi-polynomially secure), then A only asks a polynomial (resp., quasi-polynomial) number of queries. For every such oracle-aided A, with measure one over the sampling of the idealized oracle $I \xleftarrow{\$} I$, it holds that A does *not* break the security of P^I.

In the de nition above, we only require the scheme to be secure after the adversary is xed. This is along the line of the way the random oracle model is used in cryptography [BR93], and lets us easily derive certain primitives in idealized models. For example it is easy to see that a random trapdoor permutation, with measure one, is a secure TDP against any xed adversary of polynomial query complexity. Therefore, TDPs exist in the idealized model of random TDP in a black-box way. In fact stronger results are proved in the literature for other primitives. Impagliazzo and Rudich [IR89] and Gennaro and Trevisan [GT00] showed that one-way functions exist relative to the idealized model of random oracle, even if we sample the oracle rst and then go over enumerating possible attacks. Chung et al. [CLMP13] proved a similar result for collision resistant hash functions.

Remark 3.1.5. De nition 3.1.4 is di erent from saying that *P* exists in a relativized world I *I* with measure one (where we have to x the oracle rst then enumerate over all

attackers instead of fixing the attacker first). If the security condition was defined with respect to *efficient* attackers, then we could use the Borel–Cantelli lemma (see Lemma 2.2.1) then do a union bound over all such attackers, but the number of *bounded-query* attacking algorithms is not countable. Working with Definition 3.1.4 allows us to still use idealized model $\mathcal{I}$ for the purpose of proving black-box separations (where we want to say $\mathcal{P}$ is not enough to obtain another primitive $\mathcal{Q}$).

Definition 3.1.4 is called "oracle-fixed" in contrast to another "oracle-mixed" definition (stated below) in which the same security condition holds (with measure one) over the randomness of the $\mathcal{I}$ and A at the same time.

Definition 3.1.6 (Oracle-mixed constructions in idealized models)**.** We say a primitive $\mathcal{P}$ has an *oracle-mixed* black-box construction in idealized model $\mathcal{I}$ if there is an oracle-aided algorithm P such that:

> **Oracle-mixed Completeness:** $P^{\mathcal{I}}$ implements $\mathcal{P}$ correctly where the probabilities are also over $I \leftarrow \mathcal{I}$.[1] For the important case of perfect completeness, this definition is the same as oracle-fixed completeness.
>
> **Oracle-mixed Black-box Security:** Let A be an oracle-aided algorithm in idealized model $\mathcal{I}$ whose *query complexity* is bounded by the specified complexity of the attacks defined for primitive $\mathcal{P}$. We say that the oracle-mixed black-box security holds for $P^{\mathcal{I}}$ if for any such A there is a negligible $\mu(n)$ such that the advantage of A breaking $P^{\mathcal{I}}$ over the security parameter n is at most $\mu(n)$ where this bound is *also* over the randomness of $\mathcal{I}$.

Remark 3.1.7 (Oracle-fixed vs. Oracle-mixed Constructions)**.** We called the constructions of Definition 3.1.4 "oracle-fixed" because many constructions in idealized models use an "oracle-mixed" security definition. In an oracle-mixed construction P of a primitive $\mathcal{P}$ in an idealized model $\mathcal{I}$, the completeness is defined similarly to Definition 3.1.4, but when it comes to security, the advantage of A in breaking the scheme is calculated *also over the randomness of* $\mathcal{I}$. Even though oracle-fixed constructions seem to enjoy a stronger security guarantee than oracle-mixed ones, it can be shown that the oracle-fixed security does not imply oracle-mixed security when the advantage of the attack is only $1/\mathrm{poly}(n)$. For example consider a trivial primitive in the Boolean random oracle model $\mathcal{B}$ in which a trivial attacker A succeeds in its attack over security parameter n if $\mathcal{B}$ is equal to 0 over the first $\log(n)$ queries. Then the only oracle for which A succeeds in its attack for an infinite sequence of security parameters is the constant zero oracle, which has a measure zero of being sampled. However, looking ahead, we will prove in Section 7.2 the fact that when the attacker achieves constant (1) advantage over the trivial bound, an oracle-fixed black-box construction is also an oracle-mixed black-box construction.

[1] For example, an oracle-mixed construction of an ϵ-approximate IO only requires approximate correctness while the probability of approximate correctness is computed also over the probability of the input as well as the oracle.

We now present two lemmas stating that the existence of some primitive P in an idealized model I and the existence of a fully- or semi-black-box construction of Q from P would imply that Q also exists in the idealized model I.

Lemma 3.1.8 (Composition lemma)**.** *Suppose Q is a fully-black-box construction of primitive Q from primitive P, and suppose P is an oracle- xed construction for primitive P relative to I (according to De nition 3.1.4). Then Q^P is an oracle- xed implementation of Q relative to the same idealized model I.*

Proof. It is easy to see that Q^P is an implementation of Q relative to I (by completeness of the constructions P and Q), and so the completeness holds. The proof of security follows. For sake of contradiction, let A^I be any e cient query successful attacker against the implementation Q^P (of Q) in the idealized model I which rules out its oracle- xed black-box property. Namely, there is a non-zero measure fraction of $I \xleftarrow{\$} I$ for which it holds that A^I breaks the security of Q^{P^I}. For any such xed I, the security reduction $S^{A^I;I}$ (of the fully-black-box construction Q of P) would break the security of P^I. By combining the algorithms S and A we get that the e cient query attacker $(S^A)^I = B^I$ breaks the security of P^I with non-zero measure over the sampled oracle $I \xleftarrow{\$} I$. But this contradicts the assumption that P is securely realized in I in an oracle- xed black-box way. Therefore Q^P is also an *oracle- xed black-box* construction of Q relative to I. □

Lemma 3.1.9 (Composition Lemma (semi-black-box constructions))**.** *Suppose Q is a semi-black-box construction of primitive Q from primitive P, and suppose P is an oracle- xed construction for primitive P relative to I (according to De nition 3.1.4). Then Q^P is an oracle- xed implementation of Q relative to the same idealized model I.*

Proof. Let Q be the semi-black-box construction of Q from P. Let P be the implementation of P relative to I. Now, for sake of contradiction, let A^I be any e cient successful attacker against the implementation of Q^P in the idealized model I. That means by non-zero measure over the choice of $I \xleftarrow{\$} I$ it holds that A^I breaks the security of Q^{P^I}. For any such xed I, the (semi-black-box) security reduction $S^I(A)$ would break the security of P^I. This means that the attacker $S(A) = B$ breaks the security of P^I with non-zero measure over the sampled oracle $I \xleftarrow{\$} I$, which contradicts the assumption that P is securely realized in I. □

Chapter 4

Impossibility of VBB Obfuscation in Idealized Models

4.1 Introduction

Virtual Black-Box (VBB) obfuscation (see De nition 2.5.1) is a strong form of obfuscation in which the obfuscated code does not reveal any secret bit about the obfuscated program unless that information could already be obtained through a black-box access to the program. It was shown in [BGI+01, GK05] that VBB obfuscation is not possible *in general* as there is a family of functions F that could not be VBB obfuscated. Roughly speaking, F would consist of circuits C such that given any obfuscation $B = \mathsf{vO}(C)$ of C, by running B over B itself as input one can obtain a secret s about C that could not be obtained through mere black-box interaction with C. This strong impossibility result, however, did not stop researchers from exploring the possibility of VBB obfuscation for *special* classes of functions, and positive results for special cases were presented (e.g., [Can97, CMR98, DS05, Wee05, AW07, HRsV07, CD08, BC10, CRV10]) based on believable computational assumptions.

The work of Lynn, Prabhakaran and Sahai [LPS04] showed the possibility of VBB obfuscation for certain class of functions in the *random oracle model* (ROM) (see De nition 2.3.1). The work of [LPS04] left open whether general purpose obfuscator for all circuits could be obtained in the ROM or not. Note that when we allow the random oracle to be used during the obfuscation phase (and also let the obfuscated code to call the random oracle) the impossibility result of [BGI+01] no longer applies, because the proof of [BGI+01] requires the obfuscated algorithm to be a circuit in the *plain* model where no oracle is accessed. In fact, despite the impossibility of general VBB obfuscation in the plain model, a construction for VBB obfuscation in the ROM could be used as a practical heuristic obfuscation mechanism once instantiated with a strong hash function such as SHA3. This would be in analogy with the way ROM based constructions of other primitives are widely used in practice *despite* the impossibility results of [CGH04].

Canetti, Kalai, and Paneth [CKP15] proved the rst impossibility result for VBB obfuscation in a natural idealized model by ruling out the existence of general purpose VBB

obfuscators in the random oracle model, assuming the existence of trapdoor permutations. Their work resolved the open question of [LPS04] negatively. At a technical level, [CKP15] showed how to compile any VBB obfuscator in the ROM into an *approximate* VBB obfuscator in the plain model which preserves the circuit's functionality only for \most" of the inputs. This would rule out VBB obfuscation in plain model (assuming TDPs) since Bitansky and Paneth [BP13] had shown that no approximate VBB obfuscator for general circuits exist if trapdoor permutations exist.

Pass and shelat [PS16] further studied the possibility of VBB obfuscation in idealized algebraic models in which the positive results of [BGK+14, BR14] were proved. [PS16] showed that the existence of VBB obfuscation schemes in the graded encoding model (see De nition 2.3.4) highly depends on the degree of polynomials (allowed to be zero tested) in this model. In particular they showed that VBB obfuscation of general circuits is impossible in the graded encoding model of constant-degree polynomials. Their work nicely complemented the positive results of [BGK+14, BR14] that were proved in a similar (graded encoding) model but using super-constant (in fact polynomial in security parameter) polynomials.

We shall emphasize that proving *limitations* of VBB obfuscation or any other primitive in generic models of computation such as the generic group model of Shoup [Sho97] are strong lower-bounds (a la black-box separations [RTV04, IR89]) since such results show that for certain cryptographic tasks, as long as one uses certain algebraic structures (e.g., an abelian group) in a black-box way as the source of computational hardness, there will always be a *generic* attack that also treats the underlying algebraic structure in a black-box way and breaks the constructed scheme. The fact that the proposed attack is generic makes the lower-bound only stronger.

4.2 Our Results

In this chapter we extend the previous works [CKP15, PS16] on the impossibility of VBB obfuscation in idealized models of computation and generalize their results to more powerful idealized primitives. We rst focus on the generic group model of Shoup [Sho97] (see De nitions 2.3.3 and 2.3.5) and rule out the existence of general VBB obfuscation in this model for *any nite abelian group*.

Theorem 4.2.1 (Informal). *Assuming trapdoor permutations exist, there is no virtual black-box obfuscation for general circuits in the generic group model for any nite abelian group.*

The work of [PS16] implies a similar lower bound for the case of abelian groups of prime order. We build upon the techniques of [CKP15, PS16] and extend the result of [PS16] to arbitrary (even non-cyclic) nite abelian groups. See the next section for a detailed description of our techniques for proving this theorem and the next theorems described below.

We then apply our techniques designed to prove Theorem 4.2.1 to the setting of graded-encoding model studied in [PS16] and extend their results to arbitrary nite *rings* (rather

than fields) which remained open after their work. Our proof even handles *non-commutative* rings.

Theorem 4.2.2 (Informal). *Assuming trapdoor permutations exist, there is no virtual black-box obfuscation for general circuits in ideal degree-$O(1)$ graded encoding model for* any *finite ring.*

Finally, we generalize the work of [CKP15] beyond random oracles by ruling out general VBB obfuscation in random *trapdoor permutations* (TDP) oracle model. Our result extends to an arbitrary number of levels of hierarchy of trapdoors, capturing idealized version of primitives such as hierarchical identity based encryption [HL02].

Theorem 4.2.3 (Informal). *Assuming trapdoor permutations exist, there is no virtual black-box obfuscation for general circuits in the random trapdoor permutation model, even if the oracle provides an unbounded* hierarchy *of trapdoors.*

Note that the difference between the power of random oracles and random TDPs in cryptography is usually huge, as random oracle is an idealized primitive giving rise to (very efficient) *symmetric* key cryptography primitives, while TDPs could be used to construct many *public-key* objects. Our result indicates that when it comes to VBB obfuscation random TDPs are no more powerful than just random oracles.

Connection to black-box complexity of IO. Looking ahead, we note that the results of this work and those of [PS16] will be used to derive lower-bounds on the assumptions that can be used in a black-box way to construct IO. In particular, let P be a primitive implied by (i.e. exist relative to) random trapdoor permutations, generic abelian group model, or the degree-$O(1)$ graded encoding model; this includes powerful primitives such as exponentially secure TDPs or exponentially secure Diffie-Hellman-type assumptions. In Chapter 5 we show that basing IO on P in a black-box way is either impossible, or it is at least as hard as basing public-key cryptography on one-way functions (in a non-black-box way). Whether or not public-key encryption can be based on one-way functions has remained as one of the most fundamental open questions in cryptography.

4.3 Technical Overview

The high level structure of the proofs of our results follows the high level structure of [CKP15], so we start by recalling this approach. The idea is to convert the VBB obfuscator vO' in the idealized model to an *approximate* VBB obfuscation $\widehat{\mathsf{vO}}$ in the plain model which gives the correct answer $C(x)$ with high (say, $9/10$) probability over the choice of random input x and randomness of obfuscator. The final impossibility follows by applying the result of [BP13] which rules out approximate VBB in the plain model. Following [CKP15] our approximate VBB obfuscator $\widehat{\mathsf{vO}}$ in the plain model has the following high level structure.

1. **Obfuscation emulation.** Given a circuit C emulate the execution of the obfuscator vO' in the idealized model over input C to get circuit B (running in the idealized model).[1]

2. **Learning phase.** Emulate the execution of B over m random inputs for sufficiently large m. Output B and the view z of the m executions above as the obfuscated code $\widehat{B} = (B, z)$.

3. **Final execution.** Execute the obfuscated code $\widehat{B} = (B, z)$ on new random points using some form of "lazy evaluation" of the oracle while only using the transcript z of the learning phase (and not the transcript of obfuscator vO which is kept private) as the partially fixed part of the oracle. The exact solution here depends on the idealized model $\mathcal{I}$, but they all have the following form: if the answer to a new query could be "derived" from z then use this answer, otherwise generate the answer from some simple distribution.

VBB property. As argued in [CKP15], the VBB property of $\widehat{\mathsf{vO}}$ follows from the VBB property of vO and that the sequence of views z could indeed be sampled by any PPT holding B in the idealized model (by running B on m random inputs), and so it is simulatable (see Lemma 4.4.5).

Approximate correctness. The main challenge is to show the approximate correctness of the new obfuscation procedure in the plain model. The goal here is to show that if the learning phase is run for sufficiently large number of rounds, then its transcript z has enough information for emulating the next (actual) execution *consistently* with but *without* knowing the view of vO. In the case that $\mathcal{I}$ is a random oracle [CKP15] showed that it is sufficient to bound the probability of the "bad" event E that the final execution of $\widehat{B} = (B, z)$ on a random input x asks any of the "private" queries of the obfuscator vO which is not discovered during the learning phase. The work of [PS16] studies graded encoding oracle model where the obfuscated code can perform arbitrary zero-test queries for low degree polynomials $p(\cdot)$ defined over the generated labels $s_1, \ldots, s_k$. The oracle will return true if $p(s_1, \ldots, s_k) = 0$ (in which case $p(\cdot)$ is called a zero polynomial) and returns false otherwise. Due to the algebraic structure of the field here, it is no longer enough to learn the heavy queries of the obfuscated code who might now ask its oracle query $p(\cdot)$ from some "flat" distribution while its answer is correlated with previous answers.

4.3.1 Generic Group Model: Proving Theorem 4.2.1

To describe the high level ideas of the proof of our Theorem 4.2.1 it is instructive to start with the proof of [PS16] restricted to zero testing degree-1 polynomials and adapt it to the

[1] The emulation here and in next steps would require the idealized model $\mathcal{I}$ to have an efficient "lazy evaluation" procedure. For example lazy evaluation for random oracles chooses a random answer (different from previous ones) given any new query.

very close model of GGM for Z_p when p is a prime, since as noted in [PS16] when it comes to zero-testing linear functions these two models are indeed very close.[2]

Case of Z_p for prime p [PS16]. When we go to the generic group model we can ask addition and labeling queries as well. It can be shown that we do not need to generate any labels when evaluating the obfuscated code since they can be emulated using addition queries. Then, by induction, all the returned labels $t_1, \ldots, t_\ell$ for addition queries are linear combinations of the labels $s_1, \ldots, s_k$ generated during obfuscation with integer coefficients[3] and that is how we represent queries. Suppose we get an addition query $\mathbf{a} + \mathbf{b}$ and want to know the label of the group element determined by (the coefficients) $\mathbf{a} + \mathbf{b} = \mathbf{x}$. Suppose for a moment that we know s is the label for a vector of integers $\mathbf{c}$, and suppose we also know that the difference $\mathbf{x} - \mathbf{c}$ evaluates to zero. In this case, similarly to [CKP15], we can confidently return the label s as the answer to $\mathbf{a} + \mathbf{b}$. To use this idea, at any moment, let W be the space of all (zero) vectors $\boldsymbol{\alpha} - \boldsymbol{\beta}$ such that we have previously discovered same labels for $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$. Now to answer $\mathbf{a} + \mathbf{b} = \mathbf{x}$ we can go over all previously discovered labels ($\mathbf{c}$? s) and return s if $\mathbf{x} - \mathbf{c} \in \text{span}(W)$, and return a random label otherwise. The approximate correctness follows from the following two points.

> **The rank argument.** First note that if $\mathbf{x} - \mathbf{c} \in \text{span}(W)$ then the label for the vector $\mathbf{a} + \mathbf{b} = \mathbf{x}$ is indeed s. So we only need worry about cases where $\mathbf{x} - \mathbf{c} \notin \text{span}(W)$ but $\mathbf{x} - \mathbf{c}$ is still zero. The rank argument of [PS16] shows that this does not happen too often if we repeat the learning phase enough times. The main idea is that if this "bad" event happens, it increases the rank of W, but this rank can increase only k times.
>
> **Gaussian elimination.** Finally note that the test $\mathbf{x} - \mathbf{c} \notin \text{span}(W)$ can be implemented efficiently using Gaussian elimination when we work in Z_p.

Case of general cyclic groups. We first describe how to extend the above to any cyclic (abelian) group Z_m (for possibly non-prime m) as follows.

> **Alternative notion for rank of W.** Unfortunately, when we move to the ring of Z_m for non-prime m it is no longer a field and we cannot simply talk about the rank of W (or equivalently the dimension of span(W)) anymore.[4] More specifically, similarly to [PS16], we need such (polynomial) bound to argue that during most of the learning phases the set span(W) does not grow. To resolve this issue we introduce an alternative notion which here we call $\widetilde{\text{rank}}(W)$ that has the following three properties even when vectors $\mathbf{w} \in W$ are in Z_m^k (1) If $\mathbf{a} \in \text{span}(W)$ then $\widetilde{\text{rank}}(W) = \widetilde{\text{rank}}(W \cup \{\mathbf{a}\})$, and

[2] More formally, using the rank argument of [PS16] it can be shown that for the purpose of obfuscation, the two models are equivalent up to arbitrary small $1/\text{poly}(n)$ completeness error.

[3] Even though the summation is technically defined over the group elements, for simplicity we use the addition operation over the labels as well.

[4] Note that this is even the case for Z_q when q is a prime power, although finite fields have prime power sizes.

(2) if $\mathbf{a} \notin \text{span}(W)$ then $\widetilde{\text{rank}}(W) + 1 \leq \widetilde{\text{rank}}(W \cup \{\mathbf{a}\})$, and (3) $1 \leq \widetilde{\text{rank}}(W) \leq k \log |Z_m| = k \log m$. In particular in Lemma 4.4.27 we show that the quantity $\widetilde{\text{rank}}(W) := \log |\text{span}(W)|$ has these three properties. These properties together show that $\text{span}(W)$ can only "jump up" $k \log m$ (or fewer) times during the learning phase, and that property is enough to be able to apply the argument of [PS16] to show that sufficiently large number of learning phases will bound the error probability by arbitrary $1/\text{poly}(n)$.

Solving system of linear equations over Z_m. Even though m is not necessarily prime, this can still be done using a generalized method for cyclic abelian groups [McC90].

Beyond cyclic groups. Here we show how to extend the above argument beyond cyclic groups to arbitrary abelian groups. First note that to solve the Gaussian elimination algorithm for Z_m, we first interpret the integer vectors of W as vectors in Z_m^k. This "mapping" was also crucially used for bounding $\widetilde{\text{rank}}(W)$.

Mapping integers to general abelian G. When we move to a general abelian group G we again need to have a similar mapping to map W into a "finite" module. Note that we do not know how to solve these questions using integer vectors in W efficiently. In Lemma 4.4.4 we show that a generalized mapping $\rho_G(\cdot) : Z \to G$ (generalizing the mapping $\rho_{Z_m}(x) = (x \mod m)$ for Z_m) exists for general abelian groups that has the same effect; namely, without loss of generality we can first convert integer vectors in W to vectors in G^k and then work with the new W.

The alternative rank argument. After applying the transformation above over W (to map it into a subset of G^k) we can again define and use the three rank-like properties of $\widetilde{\text{rank}}(\cdot)$ (instead of $\text{rank}(W)$) described above, but here for any finite abelian group G. In particular we use $\widetilde{\text{rank}}(W) := \log |\text{span}_Z(W)|$ where $\text{span}_Z(\cdot)$ is the module spanned by W using *integer* coefficients. Note that even though G is not a ring, multiplying integers with $x \in G$ is naturally defined (see Definition 4.4.3).

System of linear equations over finite abelian groups. After the conversion step above, now we need to solve a system of linear equation $\mathbf{x}W = \mathbf{a}$ where elements of $W, \mathbf{a}$ are from G but we are still looking for *integer* vector solutions $\mathbf{x}$. After all, there is no multiplication defined over elements from G. See Section 4.4.2 where we give a reduction from this problem (for general finite abelian groups) to the case of $G = Z_m$ which is solvable in polynomial time [McC90].

4.3.2 Low-Degree Graded Encoding Model: Proving Theorem 4.2.2

To prove Theorem 4.2.3 for general finite rings, we show how to use the ideas developed for the case of general abelian generic groups discussed above and apply them to the framework

of [PS16] for low-degree graded encoding model as specified in Theorem 4.2.2. Recall that here the goal is to detect the zero polynomials by checking their membership in the module span(W). Since here we deal with polynomials over a ring (or field) R the multiplication is indeed defined. Therefore, if we already know a set of zero polynomials W and want to judge whether $\mathbf{a}$ is also (the vector corresponding to) a zero polynomial, the more natural approach is to solve a system of linear equations $\mathbf{x}W = \mathbf{a}$ over ring R.

Searching for *integer solutions* again. Unfortunately we are not aware of a polynomial time algorithm to solve $\mathbf{x}\ W = \mathbf{a}$ in general finite rings and as we mentioned above even special cases like $R = \mathbb{Z}_m$ are nontrivial [McC90]. Our idea is to try to reduce the problem back to the abelian *groups* by somehow eliminating the ring multiplication. Along this line, when we try to solve $\mathbf{x}\ W = \mathbf{a}$, we again restrict ourselves *only* to integer solutions. In other words, we do not multiply inside R anymore, yet we take advantage of the fact that the existence of integer solution to $\mathbf{x}\ W = \mathbf{a}$ is *still sufficient* to conclude $\mathbf{a}$ is a zero vector. As we mentioned above, we indeed know how to find integer solutions to such system of linear equations in polynomial time (see Section 4.4.2).

Finally note that, we can again use our alternative rank notion of $\widetilde{\text{rank}}(W)$ to show that if we run the learning phase of the obfuscation (in plain model) m times the number of executions in which $\text{span}_{\mathbb{Z}}(W)$ grows is at most poly(n) (in case of degree-$O(1)$ polynomials). This means that we can still apply the high level structure of the arguments of [PS16] for the case of finite rings *without* doing Gaussian elimination over rings.

4.3.3 Random Trapdoor Permutation Model: Proving Theorem 4.2.3.

Here we give the high-level intuition behind our result for the random TDP oracle model.

Recalling the case of Random Oracles [CKP15]. Recall the high level structure of the proof of [CKP15] for the case of random oracles described above. As we mentioned, [CKP15] showed that to prove approximate correctness it is sufficient to bound the probability of the event E that the final execution of $\widehat{B} = (B, z)$ on a random input x asks any of the queries that is asked by emulated obfuscation vO of B (let Q_O denote this set) which is *not* discovered during the learning phase. So if we let Q_E, Q_B, Q_O denote the set of queries asked, respectively, in the final execution, learning, and obfuscation phases, the bad event would be $Q_E \setminus (Q_O \cap Q_B) \neq ?$. This probability could be bound by arbitrary small $1/\text{poly}$ by running the learning phase sufficiently many times. The intuition is that all the "heavy" queries which have a $1/\text{poly}$-chance of being asked by $\widehat{B} = (B, z)$ (i.e., being in Q_E) on a random input x would be learned, and thus the remaining unlearned private queries (i.e., $Q_O \cap Q_B$) would have a sufficiently small chance of being hit by the execution of $\widehat{B} = (B, z)$ on a random input x.

Warm-up: Random Permutation Oracle. We start by first describing the proof for the simpler case of random permutation oracle. The transformation technique for the random

oracle model can be easily adapted to work in the random permutation model as follows. For starters, assume that the random permutation is only provided on one input length k; namely R: $f0,1g^k$ 7! $f0,1g^k$. If $k = O(\log \kappa)$ where κ is the security parameter, then it means that the whole oracle can be learned during the obfuscation and hardcoded in the obfuscated code, and so R cannot provide any advantage over the plain model. On the other hand if $k = \omega(\log \kappa)$ it means that the range of R is of super-polynomial size. As a result, the same exact procedure proposed in [CKP15] (that assumes R is a random oracle and shows how to securely compile out R from the construction) would also work if R is a random permutation oracle. The reason is that the whole transformation process asks poly(κ) number of queries to R and, if the result of the transformation does not work when R is a random permutation, then the whole transformation gives a poly(κ) = q query attack to distinguish between whether R is a random permutation or a random function. Such an attack cannot \succeed" with more than negligible probability when the domain of R has super-polynomial size $q^{!(1)}$ in the number of queries q[5].

Random TDP Model. Suppose $T = (G, F, F^{-1})$ is a random trapdoor permutation oracle in which G is a random permutation for generating the public index, F is the family of permutations evaluated using public index, and F^{-1} is the inverse permutation computed using the secret key (see De nition 2.3.2 for formal de nition and notation used). When the idealized oracle is $T = (G, F, F^{-1})$, we show that it is su cient to apply the same learning procedure used in the random oracle case over the *normalized* version of the obfuscated algorithm B to get a plain-model execution $\widehat{B}(x)$ that is statistically close to an execution $B^T(x)$ that uses oracle T. This, however, requires careful analysis to prove that inconsistent queries speci c to the TDP case occur with small probability.

Indeed, since the three algorithms (emulation, learning, and nal execution) are correlated, there is a possibility that the execution of B on the new random input might ask a new query that is *not* in Q_O, and yet still be inconsistent with some query in $Q_O \cap Q_B$. For example, assume we have a query q of the form $G(sk) = pk$ that was asked during the obfuscation emulation phase (and thus is in Q_O) but was missed in the learning phase (and thus is not in Q_B) and assume that a query of the form $F[pk](x) = y$ was asked during the learning phase (so it is in Q_B). Then, it is possible that during the evaluation of the circuit B, it may ask a query q' of the form $F^{-1}[sk](y)$ and since this is a new query undetermined by previously learned queries, the plain-model circuit $\widehat{B}$ will answer with some random answer y'. Note that in this case, y' would be di erent from y with very high probability, and thus even though $q \neq q'$, they are together inconsistent with respect to oracle T.

As we show in our case-by-case analysis of the learning heavy queries procedure for the case of trapdoor permutation (in Section 4.5.2), the only bad events that we need to consider (besides hitting unlearned Q_O queries, which was already shown to be unlikely) will

[5] In general, when the random permutation R is available in *all* input lengths, we can use a mixture of the above arguments by generating all the oracle queries of length $c \log(\kappa)$ (for a su ciently large constant c) during the obfuscation (in the plain model) and representing this randomness in the obfuscated circuit. This issue also exists in the trapdoor permutation and the generic group models and can be handled exactly the same way.

be those whose probability of occurring are negligible (we use the lemmas from [GKLM12] as leverage). Due to our normalization procedure, the rest of the cases will be reduced to the case of not learning heavy queries, and this event is already bounded.

4.4 Impossibility of VBB Obfuscation in Generic Algebraic Models

In this section we will formally state and prove our Theorems 4.2.1 and 4.2.2 for the generic group and graded encoding models, respectively. After going over the preliminaries related to this section and describing a useful algorithm for solving linear equations over abelian groups (which we will later use in our compilation procedure), we proceed with proving the VBB impossibility in the idealized generic group model and the graded encoding model.

4.4.1 Preliminaries

We start by stating some basic group theoretic notation, facts, and de nitions. By Z we refer to the set of integers. By Z_n we refer to the additive ring of integers modulo n. When G is an abelian group, we use + to denote the operation in G. A semigroup $(G, \ )$ consists of any set G and an associative binary operation over G.

Definition 4.4.1. For semi-groups $(G_1, \ _1), \ldots, (G_k, \ _k)$, by the direct product semi-group $(G, \ ) = (G_1 \quad G_k, \ _1 \quad _k)$ we refer to the group in which for $g = (g_1, \ldots, g_k)$ 2 $G, h = (h_1, \ldots, h_k)$ 2 G we de ne $g \quad h = (g_1 \ _1 h_1, \ldots, g_k \ _k h_k)$. If G_i's are groups, their direct product is also a group.

The following is the fundamental theorem of nitely generated abelian groups restricted to the case of nite abelian groups.

Theorem 4.4.2 (Characterization of Finite Abelian Groups)**.** *Any nite abelian group G is isomorphic to the direct product group $Z_{p_1^{\alpha_1}} \quad Z_{p_d^{\alpha_d}}$ in which p_i's are (not necessarily distinct) primes and $Z_{p_i^{\alpha_i}}$ is the cyclic group mod $p_i{}^i$.*

Definition 4.4.3 (Integer vs in-group multiplication for abelian groups)**.** For integer a 2 Z and g 2 G where G is any nite abelian group by $a \ g$ we mean adding g by itself $|a|$ times and negating it if $a < 0$. Now let g, h 2 G both be from abelian group G and let $G = Z_{p_1^{\alpha_1}} \quad Z_{p_d^{\alpha_d}}$ where p_i's are primes. If not speci ed otherwise, by $g \ h$ we mean the multiplication of g, h in G interpreted as the multiplicative semigroup that is the direct product of the *multiplicative* semigroups of $Z_{p_i^{\alpha_i}}$'s for i 2 $[d]$ (where the multiplications in $Z_{p_i^{\alpha_i}}$ are mod $p_i{}^i$).

Lemma 4.4.4 (Mapping integers to abelian groups)**.** *Let $G = Z_{p_1^{\alpha_1}} \quad Z_{p_d^{\alpha_d}}$. De ne $\rho_G : Z \to G$ as $\rho_G(a) = (a_1, \ldots, a_d)$ 2 G where $a_i = a \mod p_i{}^i$ 2 $Z_{p_i^{\alpha_i}}$. Also for $\mathbf{a} = (a_1, \ldots, a_k)$ 2 Z^k de ne $\rho_G(\mathbf{a}) = (\rho_G(a_1), \ldots, \rho_G(a_k))$. Then for any a 2 Z and g 2 $G =$*

$Z_{p_1^{\alpha_1}} \quad Z_{p_d^{\alpha_d}}$ *it still holds that* $a \quad g = \rho_G(a) \quad g$ *where the rst multiplication is done according to De nition 4.4.3, and the second multiplication is done in* G*. More generally, if* $\mathbf{a} = (a_1, \ldots, a_k)$ *2* Z^k*, and* $\mathbf{g} = (g_1, \ldots, g_k)$ *2* G*, then* $\sum_{i2[k]} a_i g_i = h\mathbf{a}, \mathbf{g}i = h\rho_G(\mathbf{a}), \mathbf{g}i$*.*

The following lemma is used in [CKP15], and here we state it in an abstract form considering only the VBB security and ignoring the completeness.

Lemma 4.4.5 (Preservation of VBB Security)**.** *Let* vO *be a PPT virtual black-box obfuscator in the I-ideal model that satis es VBB security, and let* U *be a PPT algorithm also in the I-ideal model that given input* B = $vO^I(C)$ *for some circuit* C *2* $f0,1g$ *of size* n*, outputs circuit* B^0*, and suppose* $\widehat{vO}$ *is some plain-model PPT algorithm that given circuit* C*, outputs* $\widehat{B}$*. If it holds that conditioned on any given* C*, the statistical distance between* B^0 *and* $\widehat{B}$ *are negligible, then* $\widehat{vO}$ *satis es the VBB security.*

4.4.2 Solving Linear Equations over Abelian Groups

In this section we de ne an algebraic problem relevant to our compilation procedure and we describe a polynomial time algorithm for solving it.

Definition 4.4.6. [Integer Solutions to Linear Equations over Abelian Groups (iLEAG)] Let G be a nite abelian group. Suppose we are given G (e.g., by describing its decomposition factors according to Theorem 4.4.2) an $n \quad k$ matrix A with components from G and a vector $\mathbf{b}$ *2* G^k. We want to nd an *integer* vector $\mathbf{x}$ *2* Z^n such that $\mathbf{x}A = \mathbf{b}$.

Remark 4.4.7 (Integer vs. Ring Solutions)**.** Suppose instead of searching for an integer vector solution $\mathbf{x}$ *2* Z^n we would ask to nd $\mathbf{x}$ *2* G^n and de ne multiplication in G according to De nition 4.4.3 and call this problem G-LEAG. Then any solution to iLEAG can be directly turned into a solution for G-LEAG by mapping any integer coordinate x_i of $\mathbf{x}$ into G by mapping $\rho_G(x_i)$ of Lemma 4.4.4. The converse is true also for $G = Z_n$, since any g *2* Z_n is also in Z and it holds that $\rho_G(g) = g$ *2* G. However, the converse is not true in general for general abelian groups, since there could be members of G that are not in the range of $\rho_G(Z)$. For example let $G = Z_{p^2} \quad Z_p$ for prime $p > 2$ and let $g = (2,1)$. Note that there is no integer a such that $a \mod p^2 = 2$ but $a \mod p = 1$.

Theorem 4.4.8. *There exists a polynomial time algorithm that can solve iLEAG.*

Cyclic vs General Abelian. Note that if $G = Z_p$ for a prime p then iLEAG can be solved e ciently using Gaussian elimination. If the abelian group G is cyclic Z_q, it does not matter if we de ne the variables $\mathbf{x}$ in Z^n or in Z_q^n. In this case, if q is prime, we can use the Gaussian elimination method to nd $\mathbf{x}$. Interestingly, even if q is not a prime, there is still a way to solve systems of linear equations over Z_q using a generalization of Gaussian elimination [McC90] which is the main tool we use to prove Theorem 4.4.8.

Theorem 4.4.9 ([McC90] Section 5.1)**.** *Systems of linear equations over* Z_q *for integer* q *can be solved in polynomial time. i.e., iLEAG is solvable in polynomial time for cyclic (abelian)* G*.*

In the rest of this section we prove Theorem 4.4.8. We will do so by reducing iLEAG to $G = \mathbb{Z}_q$ (where q happens to be a prime power) and applying Theorem 4.4.9.

Proof of Theorem 4.4.8 using Theorem 4.4.9. We will change iLEAG in two steps by defining simplified versions of it and reducing it to these simplified forms.

Definition 4.4.10 (Linear Equations over Arbitrary Prime Powers (LEAP))**.** Let $\{\mathbf{a}^j : j \in [m], \mathbf{a}^j \in \mathbb{Z}^n\}$ be a set of m vertical vectors in $\mathbb{Z}^n$ and suppose we are given $p_i{}^{i}$ in which p_i is a prime (but it could be that $p_i = p_j$). We want to find an integer vector $x \in \mathbb{Z}^n$ such that for all $j \in [m]$ the inner product $\langle \mathbf{x}, \mathbf{a}^j \rangle$ is zero mod $p_j{}^{j}$.

Claim 4.4.11. *Solving iLEAG can be reduced to solving LEAP in polynomial time.*

Proof. Let $A = [\mathbf{a}^1, \ldots, \mathbf{a}^k], \mathbf{b} = (b_1 \ldots, b_k)$ be a given instance of iLEAG over G. Let G be isomorphic to $\mathbb{Z}_{p_1^{\alpha_1}} \times \cdots \times \mathbb{Z}_{p_d^{\alpha_d}}$. Then we can think of every column $\mathbf{a}^i$ in A as an $n \times d$ matrix $C_i = [\mathbf{c}_i^1, \ldots, \mathbf{c}_i^d]$ for which $\langle \mathbf{x}, \mathbf{a}^i \rangle = b_i$ is equivalent to having $\langle \mathbf{x}, \mathbf{c}_i^j \rangle = 0 \mod p_j{}^{j}$ for every $j \in [d]$. And so $\mathbf{c}A = b$ will be equivalent to $\langle \mathbf{x}, \mathbf{c}_i^j \rangle = 0 \mod p_j{}^{j}$ for all $i \in [k]$ and $j \in [d]$. The latter is indeed a LEAP instance (with $k \times d$ linear constraints over a total of d different moduli). □

Definition 4.4.12 (Linear Equations over Identical Prime Powers (LEIP))**.** This problem is a special case of LEAP in which all the moduli are over the same prime, but perhaps with different powers. Namely, if $p_i{}^{i}$ and $p_j{}^{j}$ is the modulus for the i inner product, we have $p_i{}^{i} = p_j{}^{j}$ for all $i, j \in [m]$.

Claim 4.4.13. *Solving LEAP could be reduced to solving LEIP in polynomial time.*

Proof. Let m be the number of linear constraints and $p_i{}^{i}$ be the modulus for that restriction. Partition $[m]$ into sets $S_1, \ldots, S_\ell$ such that i, j are in the same partition S_k if and only if $p_i = p_j$. This partitioning defines ℓ subproblems from the original LEAP instance that we simply denote by $S_1, \ldots, S_\ell$. It is easy to see that any solution $\mathbf{x}$ to the original LEAP instance also works for all of its sub problems. We prove the converse using the Chinese reminder theorem.

Let $q_j = \max_{i \in S_j} p_i{}^{i}$ be the largest modulus whose index is in S_j. For every $i \in S_j$ let $r = q_j / p_i{}^{i}$ and then multiply $\mathbf{a}^i$, b_i, as well as the modulus restriction for them (i.e., $p_i{}^{i}$) by r. Note that this will not change the space of solutions, but will make all the modulus restrictions in S_j equal to q_j.

Now let $\mathbf{x}_j$ be the integer solution to LEIP instance S_j for $j \in [\ell]$. Using the (algorithmic proof of the) Chinese reminder theorem we can use $\mathbf{x}_j$'s to find a single integer vector $\mathbf{x}$ such that $\mathbf{x} = \mathbf{x}_j \mod q_j$ for all $j \in [\ell]$ which means $\mathbf{x}$ is an integer solution to the original LEAP instance. □

Finally note that LEIP is a special case of solving linear equations over $\mathbb{Z}_q$ for general integer q which can be solved in polynomial time according to Theorem 4.4.9.

□

Finally, we note that there is also a polynomial time algorithm for the arguably \more natural" problem G-LEAG (see Remark 4.4.7) in which we seek solutions for $\mathbf{x}A = \mathbf{b}$ *inside* ring G where the multiplication in G is de ned according to De nition 4.4.3. Note that since the multiplication of De nition 4.4.3 generalizes the notion of ring Z_n, the following theorem is a also a generalization of solving system of linear equations over Z_n, as was the case for iLEAG, but here the generalization is in a di erent direction.

Theorem 4.4.14. *G-LEAG can be solved in polynomial time for nite abelian groups.*

Proof. As opposed to the case of iLEAG we show how to reduce G-LEAG directly to LEIP. Suppose we are given an $n \quad k$ matrix A and a $1 \quad k$ vector $\mathbf{b}$ with all elements in G, and we want to nd $\mathbf{x}$ such that $\mathbf{x}A = \mathbf{b}$. Let $G = Z_{p_1^{\alpha_1}} \quad Z_{p_d^{\alpha_d}}$. Therefore all elements of $A, \mathbf{b}$ and even $\mathbf{x}$ could be thought as $1 \quad d$ vectors were the addition and multiplication in the i'th coordinate is done modulo $p_i{}^i$. Now let $(A^i, \mathbf{b}^i)$ be the LEIP instance that could be derived from $\mathbf{x}A = \mathbf{b}$ and for each element we substitute the corresponding i'th coordinate in its vector representation of G. It is easy to see that solving the G-LEAG instance is equivalent to solving *all* of the d LEIP subproblems $(A^i, \mathbf{b}^i)$ for i *2* [d] and simply using the solutions $\mathbf{x}^i$ (whose components are modulo $p_i{}^i$) to scale them up into a solution $\mathbf{x}$ in G. □

4.4.3 Generic Group Model

We start by presenting some de nitions and properties regarding the generic group model, which was formally de ned in De nition 2.3.3.

Remark 4.4.15 (Family of Groups)**.** A more general de nition allows generic oracle access to a *family* of groups *f*$G_1, G_2, \ldots$*g* in which the oracle access to each group is provided separately when the index i of G_i is also speci ed as part of the query and the size of the group G_i is known to the parties. Our negative result of Section 4.4.3 directly extends to this model as well. We use the above \single-group" de nition for sake of simplicity.

Remark 4.4.16 (Stateful vs Stateless Oracles and the Multi-Party Setting)**.** Note that in De nition 2.3.3 we used a *stateless* oracle to de ne the generic group oracle, and we separated the generic nature of the oracle itself from *how* it is used by an algorithm A^I. In previous work (e.g., Shoup's original de nition [Sho97]) a *stateful* oracle is used such that it will answer addition queries *only* if the two labels have already been *obtained* before through a labeling query to the oracle.[6]

Note that for \one party" settings in which A^I is a single algorithm, A^I \knows" the labels that it has already obtained from the oracle *I* , and so w.l.o.g. A^I would never ask any addition queries unless it has previously obtained the labels itself. However, in the multi-party setting, a party might not know the set of labels obtained by other parties. A stateful oracle in this case might reveal some information about other parties' oracle queries if the oracle does *not* answer a query (y_1, y_2) (by returning *?*) just because the labels y_1, y_2 were not previously obtained.

[6] So the oracle might return *?* even if the two labels are in the range of $\sigma(G)$.

Remark 4.4.17 (Equivalence of Two Models for Sparse Encodings)**.** If the encoding of G is sparse in the sense that $|S|/|G| = \kappa^{-\omega(1)}$ where κ is the security parameter, then the probability that any party could query a correct label before it being returned by oracle through a labeling (type 1) query is indeed negligible. So in this case any algorithm (or set of interactive algorithms) A^I would have a behavior that is statistically close to a generic algorithm that would never ask a label in an addition query unless that label is previously obtained from the oracle. Therefore, if $|S|/|G| = \kappa^{-\omega(1)}$, we can consider A^I to be an *arbitrary* algorithm (or set of interactive algorithms) in the generic group model I. The execution of A would be statistically close to a \generic execution" in which A^I never asks any label before obtaining it.

In light of Remarks 4.4.16 and 4.4.17, for simplicity of the exposition we will always assume that the encoding is sparse $|S|/|G| = \kappa^{-\omega(1)}$ and so all the generic group model algorithms are automatically (statistically close to being) generic.

Theorem 4.4.18 (Theorem 4.2.1 Formalized)**.** *Let G be any abelian group of size at most $2^{\text{poly}(\ )}$. Let* vO *be an obfuscator in the generic group model $I[G, \sigma, S]$ where the obfuscation of any circuit followed by execution of the obfuscated code (jointly) form a generic algorithm. If* vO *is an ϵ-approximate VBB obfuscator in the generic group model $I[G, \sigma, S]$ for poly-size circuits, then for any $\delta = 1/\text{poly}(\kappa)$ there exists an $(\epsilon + \delta)$-approximate VBB obfuscator $\widehat{\text{vO}}$ for poly-size circuits in the plain model.*

Remark 4.4.19 (Size of G)**.** Note that if a poly(κ)-time algorithm accesses (the labels of the elements of) some group G, it implicitly means that G is at most of exp(κ) size so that its elements could be names with poly(κ) bit strings. We chose, however, to explicitly mention this size requirement $|G| \leq 2^{\text{poly}(\ )}$ since this upper bound plays a crucial role in our proof for general abelian groups compared to the special case of nite elds.

Remark 4.4.20 (Sparse Encodings)**.** If we assume a sparse encoding i.e., $|S|/|G| = \kappa^{-\omega(1)}$ (as e.g., is the case in [PS16] and almost all prior work in generic group model) in Theorem 4.4.18 we no longer need to explicitly assume that the obfuscation followed by execution of obfuscated code are in generic form; see Remark 4.4.17.

Since [BP13] showed that (assuming TDPs) there is no $(1/2 - 1/\text{poly})$-approximate VBB obfuscator in the plain-model for general circuits, the following corollary is obtained by taking $\delta = \epsilon/2$.

Corollary 4.4.21. *If TDPs exist, then there exists no $(1/2 - \epsilon)$-approximate VBB obfuscator* vO *for general circuits in the generic group model for any $\epsilon = 1/\text{poly}(\kappa)$, any nite abelian group G and any label set S of su ciently large size $|S|/|G| = \kappa^{-\omega(1)}$. The result would hold for labeling sets S of arbitrary size if the execution of the obfuscator* vO *followed by the execution of the obfuscated circuit* vO(C) *form a generic algorithm.*

Now we formally prove Theorem 4.4.18. We will rst describe the algorithm of the obfuscator in the plain model, and then will analyze its properties.

Notation and w.l.o.g. assumptions. Using Theorem 4.4.2 w.l.o.g. we assume that our abelian group G is isomorphic to the additive direct product group $Z_{p_1^{\alpha_1}} \quad Z_{p_d^{\alpha_d}}$ where p_i's are prime. Let e_i 2 G be the vector that is 1 in the i'th coordinate and zero elsewhere. Note that $fe_1, \ldots, e_d g$ generates G. We can always assume that the rst d labels obtained by vO are the labels of $e_1, \ldots, e_d$ and these labels are explicitly passed to the obfuscated circuit B = vO(C). Let k = poly(κ) be an upper bound on the running time of the obfuscator vO for input C which in turn upper bounds the number of labels obtained during the obfuscation (including the d labels for $e_1, \ldots, e_d$). We also assume w.l.o.g. that the obfuscated code never asks any type one (i.e., labeling) oracle queries since it can use the label for $e_1, \ldots, e_d$ to obtain labels of any arbitrary g = $a_1 e_1$ + + $a_d e_d$ using a polynomial number of addition (i.e., type two) oracle queries. For $\sigma(g) = s$, a 2 Z, and $t = \sigma(a \quad g)$ we abuse the notation and denote $a \quad s = t$.

The Construction

Even though the output of the obfuscator is always an actual circuit, we nd it easier to rst describe how the obfuscator $\widehat{\text{vO}}$ generates some string $\widehat{B}$, and then we will describe how to use $\widehat{B}$ to execute the new obfuscated circuit in the plain model. For simplicity we use $\widehat{B}$ to denote the obfuscated circuit.

How to Obfuscate

The new obfuscator $\widehat{\text{vO}}$. The new obfuscator $\widehat{\text{vO}}$ uses lazy evaluation to simulate the labeling $\sigma(\ )$ oracle. For this goal, it holds a set Q of the generated labels. For any new labeling query g 2 G if $\sigma(g) = s$ is already generated it returns s. Otherwise it chooses an unused label s from S uniformly at random and adds the mapping (g ! s) to Q and returns s. For an addition query (s_1, s_2) it rst nds g_1, g_2 such that $\sigma(g_1) = s_1$ and $\sigma(g_2) = s_2$ (which exist since the algorithm that calls the oracle is in generic form) and gets $g = g_1 + g_2$. Now $\widehat{\text{vO}}$ proceeds as if g is asked as a labeling query and returns the answer. The exact steps of $\widehat{\text{vO}}$ are as follows.

1. *Emulating obfuscation.* $\widehat{\text{vO}}$ emulates $\text{vO}'(C)$ to get circuit B. Note that the type-two queries to $I[G$ 7! $S]$ could be avoided all together and be answered using labeling queries since the all the generated labels are known during the obfuscation phase.

2. *Learning phase 1 (heavy queries)*: Set Q_B = ?. For i 2 $[d]$ let $t_i = \sigma(e_i)$ be the label of e_i 2 G which is explicitly passed to B by the obfuscator vO(C) and T = $(t_1, \ldots, t_d)$ at the beginning where T will (eventually) represent the discovered labels of the obfuscation. The length of the sequence T would increase during the steps below but will never exceed k. Choose m at random from $\ell = [d3 \quad k \quad \log(jGj)/\delta e]$. For i = 1, ..., m do the following:

   Choose x_i as a random input for B. Emulate the execution of B on x_i using the (growing) set Q of partial labeling for the lazy evaluation of labels. Note that

as we said above, w.l.o.g. B only asks addition (i.e., type two) oracle queries. Suppose B (executed on x_i) needs to answer an addition query (s_1, s_2). If either of the labels $u = s_1$ or $u = s_2$ is *not* already obtained during the learning phase 1 (which means it was obtained during the initial obfuscation phase) append u to the sequence T of discovered labels by $T := (T, u)$. Using induction, it can be shown that for any addition query asked during learning phase 1, at the time of being asked, we would know that the answer to this query will be of the form $\sum_{i2[k]} a_i\ t_i$ for integers a_i. Before seeing why this is the case, let $\mathbf{a}_i = (a_{i;1}, \ldots, a_{i;k})$ be the vector of integer coe cients (of the labels $t_1, t_2, \ldots$) for the answer s that is returned to the i'th query of learning phase 1. We add $(\mathbf{a}_i\ !\ s)$ to Q_B for the returned label. To see why such vectors exist, let (s_1, s_2) be an addition query asked during this phase, and let $s\ 2\ fs_1, s_2g$. If the label s is obtained previously during learning phase 1, then the linear form $s = \sum_{i2[k]} a_i\ \ t_i$ is already stored in Q_B. On the other hand, if s is a new label discovered during an addition (i.e., type two) oracle query which just made $T = (t_1, \ldots, t_{j\ 1}, t_j = s)$ have length j, then $s = a_j\ \ t_j$ for $a_j = 1$. Finally, if the linear forms for both of (s_1, s_2) in an addition oracle query are known, the linear form for the answer s to this query would be the summation of these vectors.[7]

3. *Learning phase 2 (zero vectors)*: This step does not involve executing B anymore and only generates a set $W = W(Q_B)\quad G^k$ of polynomial size. At the beginning of this learning phase let $W = ?$. Then for all $(\mathbf{a}_1\ !\ s_1)\ 2\ Q_B$ and $(\mathbf{a}_2\ !\ s_2)\ 2\ Q_B$, if $s_1 = s_2$, let $\mathbf{a} = \mathbf{a}_1\quad \mathbf{a}_2$, and add $\rho_G(\mathbf{a})$ to W where $\rho_G(\mathbf{a})$ is de ned in Lemma 4.4.4.

4. The output of the obfuscation algorithm will be $\widehat{B} = (B, Q_B, W, T, r)$ where T is the current sequence of discovered labels $(t_1, t_2, \ldots)$ as described in Lemma 4.4.4, and r is a su ciently large sequence of random bits that will be used as needed when we run the obfuscated code $\widehat{B} = (B, Q_B, W, T, r)$ in the plain model.[8]

How to Execute

In this section we describe how to execute $\widehat{B}$ on an input x using (B, Q_B, W, T, r).[9]

Executing $\widehat{B}$. The execution of $\widehat{B} = (B, Q_B, W, T, r)$ on x will be done *identically* to to the \next" execution during the learning phase 1 of the obfuscation (as if x is the $(m + 1)$'st

[7] Note that although the sequence T grows as we proceed in learning phase 1, we already now that this sequence will not have length more than d since all of these labels that are discovered while executing the obfuscated code has to be generated by the obfuscator, due to the assumption that the sequential execution of the obfuscator followed by the obfuscated code is in generic form. Therefore we can always consider $\mathbf{a}_i$ to be of dimension k.

[8] Note that even though $W(Q_B)$ could always be derived from Q_B, and even T could be derived from an *ordered* variant of Q_B (in which the order in which Q_B has grown is preserved) we still choose to explicitly represent these elements in the obfuscated $\widehat{B}$ to ease the description of the execution of $\widehat{B}$.

[9] Note that we do not have access to the set Q_σ that was used for consistent lazy evaluation of $\sigma(\ )$.

execution of this learning phase) and even the sets $Q_B, W = W(Q_B)$ will grow as the execution proceeds, with the *only* difference described as follows.[10] Suppose we want to answer an addition (i.e., type two) oracle query (s_1, s_2) where for $b = \{1, 2\}$ we inductively know that $s_b = \sum_{i \in [k]} a_{b;i} \cdot t_i$. For $b = \{1, 2\}$ let $\mathbf{a}_b = (a_{b;1}, \ldots, a_{b;k})$ and let $\mathbf{a} = \mathbf{a}_1 + \mathbf{a}_2$.

> Do the following for all $(\mathbf{b} \to s) \in Q_B$. Let $\mathbf{c} = \mathbf{a} - \mathbf{b}$ and let $\bar{\mathbf{c}} = \rho_G(\mathbf{c}) \in G^k$ as defined in Lemma 4.4.4. Let A be a matrix whose rows consists of all vectors in W. Run the polynomial time algorithm presented in Section 4.4.2 (for $G = \mathbb{Z}_n$) to see if there is any integer solution $\mathbf{v}$ for $\mathbf{v}A = \bar{\mathbf{c}}$ as an instance of the iLEAG problem defined in Definition 4.4.6. If an integer solution $\mathbf{v}$ exists, then return s as the result (recall $(\mathbf{b} \to s) \in Q_B$), break the loop, and continue the execution of $\widehat{B}$. If the loop ended and no such $(\mathbf{b} \to s) \in Q_B$ was found, choose a random label s not in Q_B as the answer, add $(\mathbf{a} \to s)$ to Q_B and continue.

Completeness and Soundness

In this section we prove the completeness and soundness of the construction of Section 4.4.3.

Size of S. In the analysis below, we will assume w.l.o.g. that the set of labels S has superpolynomial size $|S| = \kappa^{\omega(1)}$. This would immediately hold if the labeling of G is sparse, since it would mean even $|S|/|G| \geq \kappa^{\omega(1)}$. Even if the labeling is not sparse, we will show that w.l.o.g. we can assume that G itself has super-polynomial size (which means that S will be so too). That is because otherwise all the labels in G can be obtained by the obfuscator, the obfuscated code, and the adversary and we will be back to the plain model. More formally, for this case Theorem 4.4.18 could be proved through a trivial construction in which the new obfuscator simply generates all the labels of G and plants all of them in the obfuscated code and they will be used by the obfuscated algorithm. More precisely, when the size of G (as a function of security parameter κ) is neither of polynomial size $|G| = \kappa^{O(1)}$ nor super-polynomial size $|G| = \kappa^{\omega(1)}$ we can still choose a sufficiently large polynomial $\gamma(\kappa)$ and generate all labels of G when $|G| < \gamma(\kappa)$, and otherwise use the obfuscation of Section 4.4.3.

Completeness: approximate functionality. Here we prove the the following claim.

Claim 4.4.22. *Let $\widehat{B} = (B, Q_B, W, T, r)$ be the output of the obfuscator $\widehat{\mathcal{O}}$ given input circuit C with input length α. If we run $\widehat{B}$ over a* random *input according to the algorithm described in Section 4.4.3, then it holds that*

$$\Pr_{x \in \{0,1\}^\alpha; \widehat{B} \leftarrow \widehat{\mathcal{O}}(C)} \left[\widehat{B}(x) \neq C(x)\right] \leq \Pr_{x \in \{0,1\}^\alpha; B \leftarrow \mathcal{O}^{I[G, S]}(C)} \left[B^{I[G, S]}(x) \neq C(x)\right] + \delta$$

over the randomness of $I[G, S]$, random choice of x and the randomness of the obfuscators.

[10] We even allow new labels t_i to be discovered during this execution to be appended to T, even though that would indirectly lead to an abort!

Proof. As a mental experiment, suppose we let the learning phase 1 always runs for exactly $\ell + 1 = 1 + [d3\ k\ \log(jGj)/\delta e]$ rounds but only derive the components $(Q_B, W(Q_B), T)$ based on the first m executions. Now, let x_i be the random input used in the i'th execution and y_i be the output of the i'th emulation execution the learning phase 1. Since all the executions of the learning phase 1 are perfect simulations, for every i 2 $[\ell]$, and in particular $i = m$, it holds that

$$\Pr[B^{I[G,S]}(x) \text{ 6= } C(x)] = \Pr[y_i \text{ 6= } C(x)]$$

where probability is over the choice of inputs x, x_i as well as all other randomness in the system. Thus, to prove claim 4.4.22 it will suffice to prove that

$$j \Pr[y_i \text{ 6= } C(x)] \quad \Pr[\widehat{B}(x_i) \text{ 6= } C(x)]j < \delta.$$

We will indeed do so by bounding the statistical distance between the execution of $\widehat{B}$ vs the m + 1'st execution of the learning phase 1 over the same input x_i. Here we will rely on the fact that m is chosen at random from $[\ell]$.

Claim 4.4.23. *For random* $[\ell]$ *the statistical distance between the* m + 1*'st execution of the learning phase 1 (which we call* B^0*) and the execution of* $\widehat{B}$ *over the* same *input* x_i *is* $2\delta/3$ + negl(κ)*.*

To prove the above claim we will define three type of bad events over a joint execution of $B^0 = B_{m+1}$ and $\widehat{B}$ when they are done concurrently and using the same random tapes (and even the input x_i). We will then show that (1) as long as these bad events do not happen the two executions proceed identically, and (2) the total probability of these bad events is at most $2\delta/3$ + negl(κ). In the following we suppose that the executions of B^0 and $\widehat{B}$ (over the same random input) has proceeded identically so far. Suppose we want to answer an addition (i.e., type two) oracle query (s_1, s_2) where for $b = f1, 2g$ we inductively know that $s_b = \sum_{i2[k]} a_{b;i}\ \ t_i$. Several things could happen:

> If the execution of $\widehat{B}$ finds $(\mathbf{b}\ !\ s)$ 2 Q_B such that when we take $\mathbf{c} = \mathbf{a}\ \ \mathbf{b}$ and let $\overline{\mathbf{c}} = \rho_G(\mathbf{c})$ 2 G^k and let A be a matrix whose rows are vectors in (the current) W, there is an integer solution $\mathbf{v}$ to the iLEAG instance $\mathbf{v}A = \overline{\mathbf{c}}$. If this happens the execution of $\widehat{B}$ will use s as the answer. We claim that this is the \correct" answer as B^0 would also use the same answer. This is because by the definition of W and Lemma 4.4.4 for all $\mathbf{w}$ 2 W it holds that $\mathbf{w} = (w_1, \ldots, w_k)$ is a \zero vector in G^k" in the sense that summing the (currently discovered labels in) T with coefficients $w_1, \ldots, w_k$ (and multiplication defined according to Definition 4.4.3) will be zero. As a result, $\mathbf{v}A = \overline{\mathbf{c}}$ which is a linear combination of vectors in W with integer coefficients will also be a zero vector. Finally, by another application of Lemma 4.4.4 it holds that $(c_1, \ldots, c_k) = \mathbf{c} = \mathbf{a}\ \ \mathbf{b}$ is a \zero vector in Z^k" in the sense that summing the (currently discovered labels in) T with *integer* coefficients $c_1, \ldots, c_k$ (and multiplication defined according to Definition 4.4.3) will also be zero. Therefore the answer to the query defined by vector $\mathbf{a}$ is equal to the answer defined by vector $\mathbf{b}$ which is s.

If the above does not happen (and no such $(\mathbf{b} \; ! \; s) \; 2 \; Q_B$ is found) then either of the following happens. Suppose the answer returned for (s_1, s_2) in execution of $B^{\emptyset}$ is $s^{\emptyset}$:

- **Bad event E_1:** $s^{\emptyset}$ is equal to one of the labels in Q_B. Note that in this case the executions will diverge because $\widehat{B}$ will choose a random label.
- **Bad event E_2:** $s^{\emptyset}$ is equal to one of the labels discovered in the emulation of $\mathsf{vO}^{I}(C)$ (but not present in the current Q_B).
- **Bad event E_3:** $s^{\emptyset}$ is a new label, but the label chosen by $\widehat{B}$ is one of the labels used in the emulation of $\mathsf{vO}^{I}(C)$. Note that in this case the execution of $\widehat{B}$ will not use any previously used labels in Q_B.

It is easy to see that as long as none of the events E_1, E_2, E_3 happen, the execution of $B^{\emptyset}$ and $\widehat{B}$ proceeds statistically the same. Therefore, to prove Claim 4.4.23 and so Claim 4.4.22 it is su cient to bound the probability of the events E_1, E_2, E_3 as we do below.

Claim 4.4.24. $\Pr[E_3] < \mathsf{negl}(\kappa)$.

Proof. This is because (as we described at the beginning of this subsection above) the size of S is $\kappa^{!(1)}$ but the number of labels discovered in the obfuscation phase is at most $k = \mathsf{poly}(\kappa)$. Therefore the probability that a random label from S after excluding labels in Q_B (which is also of polynomial size) hits one of at most k possible labels is $k/(|S| \; |Q_B|) = \mathsf{negl}(\kappa)$. Therefore, the probability that E_3 happens for any of the oracle queries in the execution of $\widehat{B}$ is also $\mathsf{negl}(\kappa)$. □

Claim 4.4.25. $\Pr[E_2] < \delta/(3 \log |G|) < \delta/3$.

Proof. We will prove this claim using the randomness of $m \; 2 \; [\ell]$. Note that every time that a label u is discovered in learning phase 1, this label u cannot be discovered \again", since it will be in Q_B from now on. Therefore, the number of possible indexes of $i \; 2 \; [\ell]$ such that during the i'th execution of the learning phase 1 we discover a label out of Q_B is at most k. Therefore, over the randomness of $m \quad [\ell]$ the probability that the $m + 1$'st execution discovers any new labels (generated in the obfuscation phase) is at most $k/\ell \quad \delta/(3 \log |G|)$. □

Claim 4.4.26. $\Pr[E_1] < \delta/3$.

Proof. Call $i \; 2 \; [\ell]$ a bad index if event E_1 happens conditioned on $m = i$ during the execution of $B^{\emptyset}$ (which is the $(m + 1)$'s execution of learning phase 1). Whenever E_1 happens at any moment, it means that the vector $\bar{\mathbf{c}}$ is not currently in $W(Q_B)$, *but* it *will* be added W just after this query is made. We will show (Lemma 4.4.27 below) that the size of $\mathsf{span}_{\mathsf{Z}}(W)$ will at least double after this oracle query for some set $\mathsf{span}_{\mathsf{Z}}(W)$ that depends on W and that $\mathsf{span}_{\mathsf{Z}}(W) \quad G^k$, and so $|\mathsf{span}_{\mathsf{Z}}(W)| \quad |G|^k$. As a result the number of bad indexes i will be at most $\log |G|^k = k \log |G|$. Therefore, over the randomness of $m \; 2 \; [\ell]$ the probability that $m + 1$ is a bad index is at most $k \log |G|/\ell \quad \delta/3$ □

Lemma 4.4.27. *Let $W \subseteq G^k$ for some abelian group G. Let* $\text{span}_{\mathbb{Z}}(W) = \{\sum_{\mathbf{w} \in W} a_{\mathbf{w}} \mathbf{w} \mid a_{\mathbf{w}} \in \mathbb{Z}\}$ *be the module spanned by W using integer coefficients. If $\bar{\mathbf{c}} \notin \text{span}_{\mathbb{Z}}(W)$, then it holds that*

$$|\text{span}_{\mathbb{Z}}(W \cup \{\bar{\mathbf{c}}\})| \geq 2 \cdot |\text{span}_{\mathbb{Z}}(W)|.$$

Proof. Let $A = \text{span}_{\mathbb{Z}}(W)$ and let $B = \{\bar{\mathbf{c}} + \mathbf{w} \mid \mathbf{w} \in \text{span}_{\mathbb{Z}}(W)\}$ be A shifted by $\bar{\mathbf{c}}$. It holds that $|A| = |B|$ and $A \cup B \subseteq \text{span}_{\mathbb{Z}}(W \cup \{\bar{\mathbf{c}}\})$. It also holds that $A \setminus B = \emptyset$, because otherwise then we would have $\mathbf{w} + \bar{\mathbf{c}} = \mathbf{w}'$ for some $\mathbf{w}, \mathbf{w}' \in \text{span}_{\mathbb{Z}}(W)$ which would mean $\bar{\mathbf{c}} = \mathbf{w} - \mathbf{w}' \in \text{span}_{\mathbb{Z}}(W)$ which is a contradiction. Therefore $|\text{span}_{\mathbb{Z}}(W \cup \{\bar{\mathbf{c}}\})| \geq |A| + |B| = 2 \cdot |\text{span}_{\mathbb{Z}}(W)|$ □

□

Soundness: VBB Simulatability. To derive the soundness we apply Lemma 4.4.5 as follows. vO will be the obfuscator in the ideal model and $\widehat{\text{vO}}$ will be our obfuscator in the plain model where $z' = Q_B, W, T, r$ is the extra information output by $\widehat{\text{vO}}$. The algorithm U will be a similar algorithm to $\widehat{\text{vO}}$ but only during its learning phase 1 and 2 starting from an already obfuscated B. However, U will continue generating z' using the actual oracle $I[G, S]$ instead of inventing the answers through lazy evaluation. Since the emulation of the oracle during the learning phases, and that all of Q_B, W, T, R could be obtained by only having B (and no secret information about the obfuscation phase are not needed) the algorithm U also has the properties needed for Lemma 4.4.5.

Remark 4.4.28 (General abelian vs $\mathbb{Z}_n$.)**.** Note that when $G = \mathbb{Z}_n$ is cyclic, the mapping $\rho_G : \mathbb{Z} \to G$ of Lemma 4.4.4 will be equivalent to simply mapping every $a \in \mathbb{Z}$ to $(a \mod n) \in G$. Therefore, Definition 4.4.3 generalizes the notion of $\mathbb{Z}_n$ as a ring to general abelian groups, since the multiplication $x \cdot y \mod n$ in $\mathbb{Z}_n$ is the same as a multiplication in which x is interpreted from $\mathbb{Z}$ (as in Definition 4.4.3) which is equivalent to doing the multiplication inside G according to Lemma 4.4.4.

4.4.4 Degree-$O(1)$ Graded Encoding Model

In this section we will refer to Definition 2.3.4, which is an adaptation of the definition of the Graded Encoding Model (GEM) from [PS16] restricted to the degree-d polynomials. For simplicity, as in [PS16] we also restrict ourselves to the setting in which only the obfuscator generates labels and the obfuscated code only does zero tests, but the proof directly extends to the more general setting of [BGK+14, BR14]. We also use only one finite ring R in the oracle (whose size could in fact depend on the security parameter) but our impossibility result extends to any sequence of finite rings as well.

Remark 4.4.29. Remarks 4.4.16 and 4.4.17 regarding the stateful vs stateless oracles and the sparsity of the encoding in the context of generic group model apply to the graded encoding model as well. Therefore, as long as the encoding is sparse (which is the case in Definition 2.3.4) whenever $|R|$ is of size $\kappa^{\omega(1)}$) the probability of obtaining any valid label

h = enc(v, l) through any polynomial time algorithm without it being obtained from the oracle previously (by the same party or another party) becomes negligible, and so the model remains essentially equivalent (up to negligible error) even if the oracle does not keep track of which labels are obtained previously through L_O.

We prove the following theorem generalizing a similar result by Pass and shelat [PS16] who proved this for any nite eld; here we prove the theorem for any nite *ring*.

Theorem 4.4.30. *Let R be any ring of size at most $2^{\text{poly}(\ )}$. Let* vO *be any ϵ-approximate VBB obfuscator for general circuits in the ideal degree-d graded encoding model M_R^d for $d = O(1)$ where the initialization phase of M_R^d happens during the obfuscation phase. Then for any $\delta = 1/\text{poly}(\kappa)$ there is an $(\epsilon + \delta)$-approximate obfuscator $\widehat{\text{vO}}$ for poly-size circuits in the plain model.*

As in previous sections, the following corollary is obtained from Theorem 4.4.30 by taking $\delta = \epsilon/2$.

Corollary 4.4.31. *If TDPs exist, then there exists no $(1/2 \quad \epsilon)$-approximate VBB obfuscator* vO *for general circuits in the ideal degree-d graded encoding model M_R^d for* any *nite ring R of at most exponential size $jRj \quad 2^{\text{poly}(\ )}$ and any constant degree d, assuming the initialization phase of M_R^d happens during the obfuscation phase.*

[PS16] state their theorem in a more general model where a sequence of elds of growing size are accessed. For simplicity, we state a simpli ed variant for simplicity of presentation where only one ring is accessed but we let the size of ring R to depend on the security parameter κ. Our proof follows the footsteps of [PS16] but will deviate from their approach when $R \neq \mathbb{Z}_p$ by using some of the ideas employed in Section 4.4.3.

Now we sketch the proof of Theorem 4.4.30 assuming the reader is familiar with the proof of Theorem 4.4.18 from the previous section. The high level structure of the proof remains the same.

Construction. The new obfuscator $\widehat{\text{vO}}$ will have these phases:

Emulating obfuscation. $\widehat{\text{vO}}$ emulates $\text{vO}^{M_R^d}(C)$ to get circuit B.

Learning heavy subspace of space of zero vectors: The learning phase here will be rather simpler than those of Section 4.4.3 and will be just one phase. Here we repeat the learning phase m times where m is chosen at random from $\ell = [dk \quad \log(jGj)/\delta e]$. The variables W and T will be the same as in Section 4.4.3 with the di erence that W will consist of the vector of coe cients for all polynomials whose zero test answer is true.

The returned obfuscated code will be $\widehat{B} = (B, W, T, r)$ where r is again the randomness needed to run the obfuscated code.

Executing $\widehat{B}$. To execute $\widehat{B}$ on input x, we answer zero test queries as follows. For any query vector (of coe cients) $\mathbf{a}$ we test whether $\mathbf{a} \in \mathrm{span}_Z(W)$.[11] If $\mathbf{a} \in \mathrm{span}_Z(W)$ then return true, otherwise return false.

Completeness and Soundness.

The completeness follows from the same argument given for the soundness of Construction 4.4.3. Namely, the execution of $\widehat{B}$ is identical to the execution of the m + 1's learning phase (as if it exists) up to a point where we return a wrong false answer to an answer that is indeed a zero polynomial. (Note that the converse never happens). However, when such event is about to happen, the size of $\mathrm{span}_Z(W)$ will double. Since the size of $\mathrm{span}_Z(W)$ is at most $|R|^k$, if we choose m at random from $[\ell]$ the probability of the bad event (of returning a wrong false in m + 1'st execution) is at most $k \log |R|/\ell = \delta$.

The soundness follows from Lemma 4.4.5 similarly to the way we proved the soundness of the construction of Section 4.4.3.

Extension to avoid initialization. In Theorem 4.4.30 we have a restriction which says that the initialization phase must happen during the obfuscation phase only. We can extend the proof of Theorem 4.4.30 to the case that we don't have this restriction. This entails allowing the obfuscator vO and the obfuscated circuit B to ask any type of query (be it initialization phase queries or zero-testing queries) during their execution. The reason that we can avoid this restriction is that, whenever the obfuscated circuit B asks an initialization phase query $\mathrm{enc}(v, l)$, we can treat it as a polynomial containing $v.\mathrm{enc}(1, l)$ and using that we can nd out whether we should answer this query randomly or using one of the previous labels. This is very similar to the method that we employed in the learning and execution phases of generic group model case.

Claim 4.4.32. *Let R be any ring of size at most $2^{\mathrm{poly}(\lambda)}$. Let* vO *be any ϵ-approximate VBB obfuscator for general circuits in the ideal degree-d graded encoding model M_R^d for $d = O(1)$, Then for any $\delta = 1/\mathrm{poly}(\kappa)$ there is an $(\epsilon + \delta)$-approximate obfuscator $\widehat{\mathrm{vO}}$ for poly-size circuits in the plain model.*

Proof. Suppose that obfuscated circuit is B, and let $\{h_i = \mathrm{enc}(v_i, l_i)\}_1^k$ be the obfuscator's queries. We already know that k is less than the running time of obfuscator. We might learn some pair of (h_i, v_i) during the learning phase.

Construction. The new ϵ-approximate obfuscator $\widehat{\mathrm{vO}}$ will have these phases:

Emulating obfuscation. same as previous case.

[11] Note that we do *not* solve a system of equations in R and rather search only integer solutions to $\mathbf{x}W = \mathbf{a}$ as we did in Section 4.4.3.

Learning obfuscator's queries and heavy subspace of space of zero vectors: We do exactly what we did in previous learning phase. Also if obfuscated circuit asked initialization phase queries, we memorize it.

The returned obfuscated code will be $\widehat{B}$ = (B, W, T, r) where r is again the randomness needed to run the obfuscated code.

Executing $\widehat{B}$. To execute $\widehat{B}$ on input x, we do as follows. If we saw query enc(v, l): First we check, if we memorized query enc(v, l) before, we answer it using memorized queries list otherwise we answer it randomly. Also we treat enc(v, l) as a polynomial $v.$ enc$(1, l)$. We answer zero test queries as follows. For any query vector (of coe cients) $\mathbf{a}$ we test whether $\mathbf{a}$ *2* span$_Z(W)$.[12] If $\mathbf{a}$ *2* span$_Z(W)$, return true, otherwise return false.

Completeness and Soundness.

The proof of completeness is same as previous case. The only di erence is that here we need to be sure that we answer initialization phase query correctly (call it event E). Let j_i be the index such that we saw the query enc(v_i, l_i) for the rst time. E happens if we hit one of the index j_i. Since we chose m at random, we can always bound $pr(E)$ by choosing the right l.

The soundness is same as previous case.

□

Remark 4.4.33. Note that our proof of Theorem 4.4.30 does not assume any property for the multiplication (even the associativity!) other than assuming that it is distributive. Distributivity is needed by the proof since we need to be able to conclude that the summation of the vectors of the coe cients of two zero polynomials is also the vector of the coe cients of a zero polynomial; the latter is implied by distributivity.

4.5 Impossibility of VBB Obfuscation in the random TDP Model

In this section we formally prove Theorem 4.2.3 showing that any obfuscator vO with access to a random trapdoor permutation oracle T (see De nition 2.3.2 for the de nition of the oracle) can be transformed into a new obfuscator $\widehat{\text{vO}}$ in the plain model (no access to an ideal oracle) with some loss in correctness. We start by de ning query tuples with respect to a random trapdoor permutation model followed by the formalization of Theorem 4.2.3.

[12] Note that we do *not* solve a system of equations in R and rather search only integer solutions to $\mathbf{x}W = \mathbf{a}$ as we did in Section 4.4.3.

Definition 4.5.1 (TDP query tuple)**.** Given a random TDP oracle $T = (G, F, F^{-1})$ for security parameter κ, a *TDP query tuple* consists of three query-answer pairs $(V_G, V_F, V_{F^{-1}})$ where:

$V_G = (sk, pk)$ represents a query to G on input sk and its corresponding answer pk

$V_F = ((pk, x), y)$ represents a query to $F[pk]$ on input x and its corresponding answer y

$V_{F^{-1}} = ((sk, y), x')$ represents a query to $F^{-1}[sk]$ on y and its corresponding answer x'

We say that a TDP query tuple $(V_G, V_F, V_{F^{-1}})$ is *consistent* if $x = x'$.

Definition 4.5.2 (Partial TDP query tuple)**.** A *partial* TDP query tuple is one where one or more of the elements of the tuple are unknown and we denote the missing elements with a period. For example, we say a query set Q contains a TDP query tuple $(\ , V_F, \ )$ if it contains the query-answer pair $V_F = ((pk, x), y)$ but is missing the query-answer pairs $V_G = (sk, pk)$ and $V_{F^{-1}} = ((sk, y), x')$.

Theorem 4.5.3 (Theorem 4.2.3 formalized)**.** *Let* vO *be an ϵ-approximate obfuscator for poly-size circuits in the random TDP oracle model. Then, for any $\delta = 1/\text{poly}(\kappa)$, there exists an $(\epsilon + \delta)$-approximate obfuscator* $\widehat{\text{vO}}$ *in the plain model for poly-size circuits.*

Before proving Theorem 4.5.3, we state a corollary of this theorem to rule out approximate VBB obfuscation in the ideal TDP model. Since [BP13] showed that assuming TDPs exist, $(1/2 - 1/\text{poly})$-approximate VBB obfuscator does not exist for general circuits, we obtain the following corollary by taking $\delta = \epsilon/2$.

Corollary 4.5.4. *If TDPs exist, then there exists no $(1/2 - \epsilon)$-approximate VBB obfuscator* vO *for general circuits in the ideal random TDP model for any $\epsilon = 1/\text{poly}(\kappa)$.*

The proof of Theorem 4.5.3 now follows in the next two sections. We will first describe the algorithm of the obfuscator in the plain model, and then will analyze its completeness and soundness.

4.5.1 The Construction

We first describe how the new obfuscator $\widehat{\text{vO}}$ generates some data $\widehat{B}$, and then we will show how to use $\widehat{B}$ to run the new obfuscated circuit in the plain model. We also let $l_O, l_B = \text{poly}(\kappa)$, respectively, be the number of queries asked by the obfuscator vO and the obfuscated code B to the random trapdoor permutation oracle T. Note that, for simplicity of exposition, we assume the adversary only asks the oracle for queries of size κ (i.e. the domain of the permutations in T are of fixed size κ). However, as mentioned in Section 4.3.3, we can easily extend the argument to handle $O(\log(\kappa))$-size or $\omega(\log(\kappa))$-size queries to T.

How to Obfuscate

The new obfuscator $\widehat{\text{vO}}$ in plain model. Given an ϵ-approximate obfuscator vO in the random TDP model, we construct a plain-model obfuscator $\widehat{\text{vO}}$ such that, given a circuit $C \in \{0,1\}^*$, works as follows:

1. *Emulation phase*: Emulate $\text{vO}^T(C)$. Let Q_O represent the set of queries asked by vO^T and their corresponding answers. We initialize $Q_O = ?$. For every query q asked by $\text{vO}^T(C)$, we would answer the query uniformly at random conditioned on the answers to previous queries.

2. *Canonicalize* B: Let the obfuscated circuit B be the output of $\text{vO}(C)$. Modify B so that, before asking any query of the form $F^{-1}[sk](y)$, it would first ask $G(sk)$ to get some answer pk followed by $F^{-1}[sk](y)$ to get some answer x then finally asks $F[pk](x)$ to get the expected answer y.

3. *Learning phase*: Set $Q_B = ?$. Let the number of iterations to run the learning phase be $m = 2l_B l_O/\delta$ where $l_B \quad |B|$ represents the number of queries asked by B and $l_O \quad |O|$ represents the number of queries asked by vO. For $i = \{1, ..., m\}$:

   Choose $x_i \xleftarrow{\$} D_{|C|}$

   Run $B(x_i)$. For every query q asked by $B(x_i)$:

   - If $(q, a) \in Q_O \cup Q_B$ for some answer a, answer consistently with a
   - Otherwise, answer q uniformly at random and conditioned on the answers of previous related queries in $Q_O \cup Q_B$
   - Let a be the answer to q. If $(q, a) \notin Q_B$, add the pair (q, a) to Q_B

4. The output of the obfuscation algorithm will be $\widehat{B} = (B, Q_B, R)$ where $R = \{r_1, ..., r_{|B|}\}$ is a set of (unused) oracle answers that are generated uniformly at random.

How to Execute

To execute $\widehat{B}$ on an input x using (B, Q_B, R) we simply emulate $B(x)$. For every query q asked by $B(x)$, if $(q, a) \in Q_B$ for some a then return a. Otherwise, answer randomly with one of the answers a in R and add (q, a) to Q_B.

4.5.2 Completeness and Soundness

Completeness: Approximate functionality. Consider two separate experiments (real and ideal) that construct the plain-model obfuscator exactly as described in section 4.5.1 but differ when executing $\widehat{B}$. Specifically, in the real experiment, $\widehat{B}$ emulates $B(x)$ on a random input x using Q_B and R, whereas in the ideal experiment, we execute $\widehat{B}$ and answer $B(x)$'s queries using the actual oracle T instead. In essence, in the real experiment, we can think of the execution as $B^{\widehat{T}}(x)$ where $\widehat{T}$ is the TDP oracle simulated by $\widehat{B}$ using Q_B and R as

the oracle's answers (without knowing Q_O, which is part of oracle T). We will contrast the real experiment with the ideal experiment and show that the statistical distance between these two executions is at most δ. In order to achieve this, we will identify the events that di erentiate between the executions $B^T(x)$ and $B^{\widehat{T}}(x)$, and to that end we will make use of the following two lemmas:

Lemma 4.5.5 ([GKLM12]). *Let B be a canonical oracle-aided algorithm that asks t queries to a TDP oracle T. Let E_G be the event that B asks a query of the form V_G = (sk, pk) after asking query V_F = $((pk, x), y)$, then* Pr$[E_G]$ $O(t^2/2$).

Lemma 4.5.6 ([GKLM12]). *Let B be an oracle-aided algorithm that asks t queries to a TDP oracle T and let Q be the set of queries that B have issued. Then for any new query x, the answer is either* (1) *determined completely by Q or* (2) *is drawn from a distribution with a statistical distance of $O(t/2$) away from the uniform distribution.*

Now let q be a new query that is being asked by $B^{\widehat{T}}(x)$. We present a case-by-case analysis of all possible queries to identify the cases that can cause discrepancies between the real and ideal experiments:

Case 1: If q is determined by the queries in Q_B in the real experiment then it is also determined by Q_B in the ideal experiment.

Case 2: If q is not determined by Q_B [Q_O in the ideal experiment then it is also not determined by Q_B in the real experiment. In the ideal experiment the query will be answered randomly and consistently with respect to Q_B [Q_O whereas in the real experiment the query will be answered randomly and consistently with respect to Q_B. By Lemma 4.5.6, the answers will be from a distribution that is statistically close to uniform.

Case 3: If q is not determined by Q_B in the real experiment then, depending on the queries in Q_O, it may or may not be so the ideal experiment:

- **Case 3a**: The query q is in Q_O. In that case, in the real experiment, the answer would be random whereas in the ideal experiment it would use the correct answer from Q_O.
- **Case 3b**: The query q is of type V_G = (sk, pk) and the corresponding partial TDP query tuple $(., V_F, V_{F\ 1})$ is in Q_O
- **Case 3c**: The query q is of type V_F = $((pk, x), y)$ and the corresponding partial TDP query tuple $(V_G, ., V_{F\ 1})$ is in Q_O
- **Case 3d**: The query q is of type $V_{F\ 1}$ = $((sk, y), x)$ and the corresponding partial TDP query tuple $(V_G, V_F, .)$ is in Q_O
- **Case 3e**: The query q is of type V_F = $((pk, x), y)$ and V_G = (sk, pk) is in Q_B, but $V_{F\ 1}$ = $((sk, y), x)$ is in Q_O

- **Case 3f**: The query q is of type $V_{F\ 1} = ((sk, y), x)$ and $V_G = (sk, pk)$ is in Q_B, but $V_F = ((pk, x), y)$ is in Q_O
- **Case 3g**: The query q is of type $V_F = ((pk, x), y)$ and $V_{F\ 1} = ((sk, y), x)$ is in Q_B, but $V_G = (sk, pk)$ is in Q_O
- **Case 3h**: The query q is of type $V_{F\ 1} = ((sk, y), x)$ and $V_F = ((pk, x), y)$ is in Q_B, but $V_G = (sk, pk)$ is in Q_O

We note that the bad events that can cause any di erences between the real and ideal experiments are case 2 and parts of case 3. For case 2, Lemma 4.5.6 ensures that this event happens with negligible probability. For case 3a, learning heavy queries would diminish the e ect of this event. For cases 3$b, 3e$, and 3f, Lemma 4.5.5 ensures that this event happens with negligible probability since V_G was issued *after* V_F and/or $V_{F\ 1}$ was asked. For cases 3c and 3d, the remaining query from the tuple would have been de ned in Q_O and is thus captured during the learning of heavy queries. For case 3g, if V_G and $V_{F\ 1}$ were asked during the emulation or learning phases, then V_F would also be de ned and thus can be learned. However, if $V_{F\ 1}$ was asked during the execution phase then, due the canonicalization of B, it would have to ask $V_G \in Q_O$ which reduces to case 3a. Similarly, for case 3h, due the canonicalization of B, we would have to ask $V_G \in Q_O$ and this reduces to case 3a once again.

For any x, de ne $E_k(x)$ to be the event that case k happens and let event $E(x) = (E_2(x) \_ E_{3a}(x) \_ E_{3b}(x) \_ E_{3e}(x) \_ E_{3f}(x))$. Assuming that event E does not happen, the output distributions of $B^T(x)$ and $B^{\widehat{T}}(x)$ are identical. More formally, the probability of correctness for $\widehat{\mathsf{vO}}$ is:

$$\Pr_x[B^{\widehat{T}}(x) \in C(x)] = \Pr_x[B^{\widehat{T}}(x) \in C(x) \wedge \neg E(x)] + \Pr_x[B^{\widehat{T}}(x) \in C(x) \wedge E_1(x)]$$
$$\Pr_x[B^{\widehat{T}}(x) \in C(x) \wedge \neg E(x)] + \Pr_x[E(x)]$$

By the approximate functionality of vO, we have that:

$$\Pr_x[\mathsf{vO}^T(C)(x) \in C(x)] = \Pr_x[B^T(x) \in C(x)] \quad \epsilon(\kappa)$$

Therefore,

$$\Pr_x[B^{\widehat{T}}(x) \in C(x) \wedge \neg E(x)] = \Pr_x[B^T(x) \in C(x) \wedge \neg E(x)] \quad \epsilon$$

We are thus left to show that $\Pr[E(x)] \quad \delta$. By Lemma 4.5.6, $\Pr[E_2(x)] \quad \mathrm{negl}(\kappa)$ and by Lemma 4.5.5, $\Pr[E_{3b} \_ E_{3e}(x) \_ E_{3f}(x)] \quad \mathrm{negl}(\kappa)$ via a union bound. The probability of event E_{3a} was already given in [CKP15], but for the sake of completeness we show our version of the analysis here. As a result, we get that $\Pr[E(x)] \quad \delta/2 + \mathrm{negl}(\kappa) \quad \delta$.

Claim 4.5.7. *It holds that* $\Pr_x[E_{3a}(x)] \quad \delta/2$.

Proof. Let $(q_1, ..., q_{l_B})$ be the sequence of queries asked by $B^{\widehat{T}}(x)$ where $l_B \quad |B|$, and let $q_{i;j}$ be the j^{th} query that is asked by $B^T(x_i)$ during the i^{th} iteration of the learning phase. We de ne $E^j_{3a}(x)$ to be the event that the j^{th} query of $B(x)$ is in Q_O but not in Q_B. We also

de ne $p_{q;j}$ to be the probability that $q_j = q$ for any query q and $j \in [l_B]$. We can then write the probability of E_{3a} as follows:

$$\begin{aligned}
\Pr_x[E_{3a}(x)] &\leq \Pr_x[E^1_{3a}(x) \wedge \ldots \wedge E^{l_B}_{3a}(x)] \\
&= \sum_{j=1}^{l_B} \Pr_x[\neg E^1_{3a}(x) \wedge \ldots \wedge \neg E^{j-1}_{3a}(x) \wedge E^j_{3a}(x)] \\
&\leq \sum_{j=1}^{l_B} \sum_{q \in Q_O} \Pr_x[q_j = q \wedge (q_{1;j} \neq q \wedge \ldots \wedge q_{m;j} \neq q)] \\
&= \sum_{j=1}^{l_B} \sum_{q \in Q_O} p_{q;j}(1 - p_{q;j})^m \leq \sum_{j=1}^{l_B} \sum_{q \in Q_O} \frac{1}{m} \leq \sum_{j=1}^{l_B} \frac{l_O}{m} = \frac{l_B l_O}{m}.
\end{aligned}$$

Thus, given that $m = 2l_B l_O/\delta$, we get $\Pr[E_{3a}(x)] \leq \delta/2$. □

Soundness: VBB Simulatability. To show that the security property is satis ed, it su ces to provide a PPT algorithm U^T in the ideal TDP model that takes as input $\mathsf{vO}^T(C)$ for some circuit C and outputs a distribution that is statistically close to the output distribution of $\widehat{\mathsf{vO}}$. If that is the case, we can invoke Lemma 4.4.5 and conclude that $\widehat{\mathsf{vO}}$ is also VBB-secure.

The description of U is precisely the same as Steps 2-4 of the procedure detailed in Section 4.5.1 except that queries made by $B = \mathsf{vO}^T(C)$ are answered using oracle T instead of being randomly simulated. If we let (B, Q_B, R) be the output of $U^T(\mathsf{vO}^T(C))$ then we can easily see that it is identically distributed to the output distribution of $\widehat{\mathsf{vO}}$ since, in both cases, Q_B has query-answers with consistent and random TDP query tuples. They di er only by how these query answers are generated (U^T answers them using T, while $\widehat{\mathsf{vO}}$ simulates them using lazy evaluation with respect to some oracle $\widehat{T}$ distributed the same as T).

4.5.3 Extension to hierarchical random TDP

In this section, we reason that the proof for the ideal TDP case can be extended to hierarchical TDP oracles as well. We start by de ning how the oracle for the random hierarchical trapdoor permutation primitive changes from De nition 2.3.2.

Definition 4.5.8 (Random Hierarchical Injective Trapdoor Functions)**.** For any security parameter n and $l = \mathrm{poly}(\kappa)$, an l-level random *hierarchical injective trapdoor function* (HTDF) oracle T^l consists of $2l + 3$ subroutines $(\{J_i\}_{i=1}^{l+1}, \{K_i\}_{i=0}^{l+1})$ de ned as follows:

> $K_i[f_{i-2}, id_{i-1}](td_i)$: An injective function, indexed by identity vector $f_{i-2} = (id_0, \ldots, id_{i-2})$ and id_{i-1}, that accepts as input an i-level trapdoor $td_i \in \{0,1\}^m$ and outputs a randomly chosen identity $id_i \in \{0,1\}^n$ where $m = 10\kappa l$ if $i \in [1, l]$ and $m = \kappa$ (i.e. it is a permutation) if $i = \{0, l+1\}$.

$J_i[f_{i\ 2}, td_{i\ 1}](id_i)$: An injective function, indexed by identity vector $f_{i\ 2} = (id_0, ..., id_{i\ 2})$ and $td_{i\ 1}$ that, given the identity $id_i \in \{0,1\}^*$, outputs the corresponding trapdoor $td_i \in \{0,1\}^m$ where $m = 10\kappa l$ if $i \in [1, l]$ and $m = \kappa$ (i.e. it is a permutation) if $i = \{0, l+1\}$.

Note that, for any fixed $f_{i\ 2}$, if $td_i = J_i[f_{i\ 2}, td_{i\ 1}](id_i)$ and $id_{i\ 1} = K_{i\ 1}[f_{i\ 3}, id_{i\ 2}](td_{i\ 1})$ then $id_i = K_i[f_{i\ 2}, id_{i\ 1}](td_i)$. In other words, we can think of K_i as the inverse of J_i only if the indices of the two functions match (that is, the trapdoor $td_{i\ 1}$ indexing J_i corresponds to the identity $id_{i\ 1}$ indexing K_i).

Remark 4.5.9. It is also crucial to note that we used (sparse) injective functions for generating the intermediate levels of trapdoor. Such a change was made in order to obtain interesting primitives from this oracle, such as fully-secure hierarchical identity-based encryption (HIBE). If permutations were used instead, we would only achieve HIBE with security against adversaries that *do not* choose an identity for the permutation F to attack. Furthermore, removing K_i for $i \in [1, l]$ as a way to prevent this attack's capability hinders our ability to perform the canonicalization procedure for the obfuscated circuit.

Remark 4.5.10. For the special case of 1-level HTDF (i.e. TDP), we only have three permutations: $K_0, K_1[id_0]$ and $J_1[td_0]$, which correspond to permutations $G, F[pk]$, and $F^{-1}[sk]$, respectively in the language of TDP that we used in De nition 2.3.2. Note that here, we would refer to 0-level identities as master public keys and 0-level trapdoors as master secret keys.

We also present a variant of TDP query tuples that generalizes De nition 4.5.1 to work with hierarchical injective trapdoor functions.

Definition 4.5.11 (HTDF query tuple)**.** Given a random l-level HTDF oracle given as $T^l = (\{J_i\}_{i=1}^{l+1}, \{K_i\}_{i=0}^{l+1})$, an i-level *HTDF query tuple* consists of three (possibly) related query-answer pairs $(V_{K_{i\ 1}}, V_{K_i}, V_{J_i})$ where, for any fixed $f_{i\ 2} = (id_0, ..., id_{i\ 2})$:

$V_{K_{i\ 1}} = (td_{i\ 1}, id_{i\ 1})$ represents a query to $K_{i\ 1}[f_{i\ 3}, id_{i\ 2}]$ on input $td_{i\ 1}$ and its corresponding answer $id_{i\ 1}$

$V_{K_i} = ((id_{i\ 1}, td_i), id_i)$ represents a query to $K_i[f_{i\ 2}, id_{i\ 1}]$ on input td_i and its corresponding answer id_i

$V_{J_i} = ((td_{i\ 1}, id_i), td_i')$ represents a query to $J_i[f_{i\ 2}, td_{i\ 1}]$ on input id_i and its corresponding answer td_i'

We say that an i-level HTDF query tuple is *consistent* if $td_i = td_i'$.

Remark 4.5.12. For the purposes of comparison, we note that, for the special case of 1-level HTDF (i.e. TDP), we only have TDP query tuples of the form $(V_{K_0}, V_{K_1}, V_{J_1}) = (V_G, V_F, V_{F^{-1}})$. Thus, $V_G = (sk, pk)$ represents a query to G on $sk = td_0$ and the answer $pk = id_0$, $V_F = ((pk, x), y)$ represents a query to F_{pk} on $x = td_1$ and the answer $y = id_1$, and $V_{F^{-1}} = ((sk, y), x')$ represents a query to F_{sk}^{-1} on y and the answer x', which should be x if the tuple is consistent.

Extension of the proof. The extension of the impossibility result to random HTDF is straightforward, so we will outline the main di erences between the TDP case and describe how to resolve the issues that are related to this oracle. First, we still perform the normalisation procedure on $\widehat{\mathsf{vO}}$ and B where the query behaviour of these algorithms are modi ed such that for any query q of the form $J_i[f_{i\ 2}, td_{i\ 1}](id_i)$, we rst ask $K_{i\ 1}[f_{i\ 3}, id_{i\ 2}](td_{i\ 1})$ to get $id_{i\ 1}$. This allows us to discover whether we have a query $K_i[f_{i\ 2}, id_{i\ 1}](td_i)$ whose answer is id_i, in which case we can answer q with td_i. This procedure ensures that all query tuples that contain J_i queries are consistent.

We now turn to verifying whether the proof of approximate functionality for TDP holds in this case as well and, in particular, focus on the event $E(x)$ that was de ned Section 4.5.2. The main issue that we have to consider, which is unique to the HTDF case, is the possibility that di erent consistent TDP query tuples can be related to each other, and an overlap between these queries may cause an inconsistency in one of the tuples. Speci cally, an i-level TDP query tuple of the form $(V_{K_{i\ 1}}, ,)$ might overlap with an $(i \quad 1)$-level TDP query tuple $(, , V_{J_{i\ 1}})$ from Q_O, where the answer of $V_{K_{i\ 1}}$ is inconsistent with that of $V_{J_{i\ 1}}$. However, our normalisation procedure prevents precisely this issue as any TDP query tuple that contains $V_{J_{i\ 1}}$ must also have $V_{K_{i\ 1}}$, which means that the queries should not overlap otherwise event E_1 occurs leading to a contradiction to our initial assumption.

Chapter 5

Separating IO from Standard Assumptions

5.1 Introduction

In this chapter we begin by describing our first set of impossibility results for IO. We start by showing a lower-bound for IO that holds for any primitive implied by a random oracle (e.g., exponentially secure one-way functions or collision-resistant hash functions) and applies to *fully* black-box constructions that treat the primitive and the adversary in a black-box way (see Definition 3.1.2). We then show a hardness result for semi-black-box constructions of IO from any primitive implied by the random trapdoor permutation oracle or the constant-degree graded encoding model. Finally, we prove *fully black-box separations* of IO from any primitive implied by the random trapdoor permutation oracle or the constant-degree graded encoding model (thus improving upon the second result).

In this section, we discuss at a high level the ideas behind each result then dedicate a separate section for each result to discuss the details and full proofs.

5.1.1 Technical Overview: Separating IO from the Random-Oracle

Our first lower bound proves that any primitive implied by a random oracle does not imply an IO scheme with perfect completeness in a black-box way.

Theorem 5.1.1 (Fully black-box separation from primitives implied by random oracle)**.** *If* $\mathbf{NP} \not\subseteq \mathbf{co\text{-}NP}$*, there is no fully black-box construction of perfectly complete IO from collision-resistant hash functions or more generally any primitive implied by a random oracle in a black-box way.*

Intuition behind the proof. To prove Theorem 5.1.1 we first prove a useful lemma (Lemma 5.2.2) which, roughly speaking, asserts that for any pair of circuits C_1, C_2, either (1) a (computationally unbounded) polynomial-query attacker can guess which one is obfuscated in the random oracle model with a probability close to one, or that (2) there is a way to

obfuscate them into the same output circuit B. The latter can be used as a witness that C_1 and C_2 compute the same function, assuming that the obfuscation is an IO. Now consider the set of equivalent and same-size circuit $C = f(C_0, C_1) \; j \; C_0 \quad C_1 \wedge |C_0| = |C_1|g$. If Case (1) happens for an in nite subset of C, we get a poly-query attacker against iO in the random oracle model which is su cient for deriving the black-box separations of Theorem 5.1.1. On the other hand, if Case (1) happens only for a nite subset of C, we get an e cient procedure to certify the equivalence of two given circuits, implying $\mathbf{NP} \not\subseteq \mathbf{P}$. We prove Lemma 5.2.2 by reducing it to a result by Mahmoody and Pass [MP12] who ruled out the existence of non-interactive commitments from one-way functions. To do so, we construct a non-interactive commitment scheme based on common input (C_1, C_2) in the random oracle model, and we show that: the cheating receiver strategy of [MP12] implies our Case (1), and the cheating sender strategy of [MP12] implies our Case (2). The result of [MP12] shows that either of these strategies always exist. See Section 5.2 for the formal proof.

5.1.2 Technical Overview: Hardness of Semi-Black-Box Constructions of IO

Our second lower bound does not rule out black-box construction of IO, but rather shows that achieving such constructions from a large variety of primitives is as hard as the long standing open question of basing public-key encryption on one-way functions. It also captures a larger class of security reductions known as semi-black-box reductions [RTV04] that allow the security reduction to access the adversary in a non-black-box way (see De nition 3.1.3).

Theorem 5.1.2 (Hardness of semi-black-box construction)**.** *Let P be a primitive that provably exists relative to the random trapdoor permutation oracle, the generic group model (for any nite abelian group) or the degree-$O(1)$ graded encoding model (for any nite ring). Any semi-black-box construction of IO from P (constructively) implies a construction of semantically secure public-key encryption from one-way functions.*

Primitives captured by Theorem 5.1.2. Theorem 5.1.2 captures a large set of powerful cryptographic primitives that could be constructed in idealized models. For example trapdoor permutations (and any primitive implied by TDPs in a black-box way) trivially exist relative to the idealized model of random TDPs. Other primitives that exist in this model include such powerful objects as non-interactive zero-knowledge proofs for $\mathbf{NP}$ [BY93, Gol11]. Even primitives that we do not know how to construct from TDPs in a black-box way (e.g., CCA secure public key encryption) are known to exist in the random TDP model [BR93]. The generic group model de ned by Shoup [Sho97] (see De nition 2.3.3) is an idealized model in which (a black-box form of) the DDH assumption holds unconditionally. Therefore, our separation of Theorem 5.1.2 covers any primitive that could be constructed from DDH assumption in a black-box way. The same holds for *bilinear* assumptions in the graded encoding model (see De nition 2.3.4) of degree 2. Namely, primitives that could be constructed from bilinear assumptions (in a black-box way) exist in the degree-$O(1)$ graded

encoding model unconditionally. This includes one-round 3-party key-agreement [Jou00], (hierarchical) identity based encryption [BF01, GS02, HL02], etc.

Intuition behind the proof. Our main tool in proving Theorem 5.1.2 is the following theorem which is an implicit consequence of the techniques used in Chapter 4. Even though the focus of that chapter was on virtual black-box obfuscation, the same construction presented there for the case of VBB implies the following theorem for IO.

Theorem 5.1.3 (Informal). *The existence of IO in any of the idealized models of: random trapdoor permutation oracle, generic group model for nite abelian groups, or the degree-$O(1)$ graded encoding model for nite rings, implies $1/p(\kappa)$-approximate IO in the plain model for any polynomial $p(\kappa)$.*

We then show that the existence of $(1/6)$-approximate IO and any one-way functions imply the existence of \approximately correct" and \approximately secure" public-key encryption schemes. In order to prove this we employ a construction of Sahai and Waters [SW14] in which they show that IO and OWF imply PKE. Here we show that the same construction, when instantiated using *approximate* IO, leads to \approximately correct" and \approximately secure" public-key encryption. Finally we use Holenstein [Hol06] result to amplify any approximately-correct and approximately-secure PKE into a standard semantically secure PKE for su ciently good approximation. See Section 5.3 for the details.

Remark 5.1.4. Our proof of Theorem 5.1.1 relies on perfect completeness of IO. Theorem 5.1.2 above holds even if with negligible probability over the obfuscator's randomness the obfuscated circuit does not compute the same function. We will later develop more techniques extending Theorem 5.1.1 to allow negligible error over the randomness of the obfuscator.

Previous work on *hardness* of black-box constructions. Theorem 5.1.2 has the same spirit as the result by Impagliazzo and Rudich [IR89] who show that any semi-black-box construction of key agreement from one-way functions would imply $\mathbf{P} \neq \mathbf{NP}$. Therefore, the fact that we are far from proving $\mathbf{P} \neq \mathbf{NP}$ implies that we as far from basing key agreements on one-way functions in a black-box way.[1] Similarly, our Theorem 5.1.2 shows that as long as we are not able to base public-key encryption on one-way functions, we cannot base IO on a variety of strong primitives in a semi-black-box way. Other results of the same avor exist in connection with program checkers [BK95] for $\mathbf{NP}$. Mahmoody and Xiao [MX10] showed that any construction of one-way functions based on worst-case hardness of $\mathbf{NP}$ implies program checkers for $\mathbf{NP}$ whose existence is one of the long standing open questions in complexity theory.

[1] Formalizing semi-black-box constructions interpreting the result of [IR89] in this context is due to [RTV04].

Falsifiability of IO. An intriguing open question regarding assumption complexity of IO is whether IO can be based on a \falsi able" assumption [Nao03]. A falsi able assumption is one with an e cient challenger security game. The question arises because an adversary attacking an IO scheme starts by proposing two *equivalent* circuits but an e cient challenger has no direct way to verify the equivalence. Since our primitives used in the theorems above are falsi able, a separation of IO from falsi able assumptions would imply our results for the case of *polynomially secure* primitives. However, constructions of IO based on exponentially secure falsi able assumptions are indeed known [PST14]. Therefore, our results are interesting even if one can prove that IO cannot be based on falsi able assumptions. Moreover, the known lower bounds against falsi able assumptions [Pas11, GW11] are proved only for *black-box* proofs of security in which the adversary is used in a black-box way. Our Theorem 5.1.2 holds even for semi-black-box constructions in which the security reduction could use the adversary in a non-black-box manner.

5.1.3 Technical Overview: Separating IO from TDP and Constant-Degree GEM

By relying on the recent elegant work of Brakerski, Brzuska, and Fleischhacker [BBF16] in which they rule out *statistically secure approximately correct* IO (based on standard assumptions) we can improve upon Theorem 5.1.2 and rule out fully black-box constructions of IO from all the primitives of Theorem 5.1.2 (based on the same assumptions used in [BBF16]). Namely, the following holds:

Theorem 5.1.5. *Let P be a primitive that provably exists relative to random trapdoor permutation oracle, the generic group model (for any nite abelian group) or the degree-$O(1)$ graded encoding model (for any nite ring) in a way that it is secure against any \bounded-query" (computationally unbounded) attacker with probability (measure) one. Assuming the existence of one-way functions and that $\mathbf{NP} \not\subseteq \mathbf{coAM}$ (which is true if the polynomial-time hierarchy does not collapse) there is no fully black-box construction of IO from any such primitives P.*

Even though Theorem 5.1.2 is potentially applicable to more primitives, all the primitives that we listed after Theorem 5.1.2 which t into the requirements of Theorem 5.1.2 have black-box proofs in their corresponding idealized models, and thus Theorem 5.1.5 applies to them as well.

Intuition behind the proof. Recall that a fully black-box construction of a primitive *Q* from another primitive *P* consists of two oracle PPT algorithms (Q, S) such that Q^P implements *Q* given access to any oracle P that implements *P*, and $S^{P;A}$ turns any oracle attacker A against Q^P into an attack against P itself (see De nition 3.1.2 for more details).

Similar to the rst result (and in general for any black-box separation), our proof that a primitive *P* does not imply IO in a black-box way presents a polynomial-query attacker A that breaks the security of any IO construction iO in an idealized model *I* that provides an

\unquestionably secure" instantiation P (against computationally unbounded polynomial-query attackers).Intuitively, the existence of such an A rules out the possibility of a fully black-box construction construction (iO, S) of IO from P by simple *composition*. First, the construction $iO^{P^I} = (iO^P)^I$ yields an implementation of IO in the same idealized model I. But attacker A breaks every such construction of IO and therefore the security reduction $S^{P^I;A^I}$ implies the existence of a new attacker $(S^A)^I$ that calls the idealized oracle I a polynomial number of times and breaks the implementation P^I of P. But this leads to a contradiction in this model which is a contradiction. That is because P^I is an \unquestionably secure" construction of P in I. Based on this rather informal argument, it seems what we need is just a poly-query attack against any implementation of IO in idealized models.

The recent work of Brakerski et al. [BBF16], when combined with the techniques of Chapter 4 show an attacker that can break any IO scheme in either of the idealized model I of random trapdoor permutations and degree-$O(1)$ graded encoding models by asking a polynomial number of oracle queries. In particular, the techniques of Chapter 4 show how to \compile out" the idealized oracle I from the IO scheme and achieve an *approximately-correct* IO scheme in the plain model that is correct on, say, $99/100$ of the input points. Brakerksi et al. then show that any such approximately-correct IO scheme can be broken by a computationally unbounded attacker.[2] As pointed out in [BBF16], this means that any IO scheme will be broken in the idealized model I, and in particular the computationally unbounded attacker B can be modi ed into a computationally unbounded, yet *polynomial-query* attacker A against the original IO in the idealized model I.

At a rst glance, it seems that the attacker A of [BBF16] against IO in an idealized model I would immediately imply the desired black-box separation between IO and primitives that exist in model I. However, the challenge, roughly speaking, is that this attacker of does not succeed in breaking IO with probability close to 1, and doing so is left as an open question. In order to see the challenge more clearly, we need to further discuss the *big picture* argument above and see how an attack in the idealized model I exactly implies the black-box separation.

A crucial point is that to apply the security reduction $S^{P^I;A^I}$ and get the desired attack against P^I, we must x the oracles P^I and A^I into deterministic functions (which requires us to sample and x I) because only then is S guaranteed to generate an attack. However, while xing I, we want to keep the promise that A^I is still a \successful" attack. Handling both tasks simultaneously may raise an issue because all attacks in idealized models and in particular the attack against IO in idealized models that is implied by [BBF16] are successful with probability taken over the randomness of the idealized oracle I.

Here is where the Borel-Cantelli lemma (Lemma 2.2.1) usually comes to help, but only if the attack succeeds with high probability. In particular, if the demonstrated attacker A wins the security game for security parameter κ with probability e.g., $1 - 1/\kappa^4$, then by an averaging argument, with probability at least $1 - 1/\kappa^2$ over the sampled oracle I, A successfully attacks the game on security parameter κ. Therefore, since the probability of

[2] The attack of [BBF16] assumes the existence of OWFs and that $\mathbf{NP} \not\subseteq \mathbf{coAM}$, and that is where we get these assumptions for our separation as well.

the \fail" event is $\sum_{=1}^{1} 1/\kappa^2 = O(1)$, Borel-Cantelli lemma implies that with measure one over[3] the sampled oracle I it holds that A is a successful attack for all but nitely many security parameters.

By the above discussion on how to use Borel-Cantelli, we would be done if the attacker of [BBF16] succeeds with probability $1 - 1/\text{poly}(\kappa)$. However, their attack works against (ϵ, δ) statistical approximate[4] IO when $2\epsilon + 3\delta < 1$; thus, by making optimal parameter choices, their attacker only succeeds (in guessing the obfuscated circuit) with probability $1/2 + 1/6$ which is not arbitrarily close to 1. As a result, when combined with the compilation techniques of Chapter 4, we would only get an attack against IO in idealized models that succeeds with *some constant* advantage over $1/2$ (and thus fails with some constant probability). Thus, we can no longer apply the Borel-Cantelli lemma as we did before because the summation of the probability of failure becomes unbounded. Thus, we can no longer conclude that this attack would remain successful for an in nite sequence of security parameters κ [5] when we sample and x the idealized oracle I. In fact, there are examples of protocols in idealized models with attacks against them with $1/\text{poly}(\kappa)$ advantage over the trivial bound, but once the randomized oracle is sampled and xed, they do *not* remain successful over an in nite sequence of security parameters (see Remark 3.1.7).

To overcome this issue, we provide a variant of the Borel-Cantelli lemma (see Lemma 2.2.2) which allows us to make su ciently strong conclusions about the attacker as long as the attacker A succeeds with a *constant* advantage over the trivial bound. Note that Borel-Cantelli (when applicable) would imply a stronger result, because it shows that the attack will remain successful for *all but nitely many* security parameters, while our lemma shows that it only succeeds for an in nite sequence of security parameters. However, even this weaker conclusion is still enough for the security reduction $S^{P^I;A^I}$ to be able to use A and give a polynomial-query attack against P^I.

The scope of this argument does not seem to be at all limited to proving separations for IO, and we believe that it could potentially be applied to other primitives as well. Namely, it shows that to derive a black-box separation of Q from P, it is enough to break Q in an idealized model that gives P by asking a polynomial number of queries and a *constant* advantage over the trivial bound.

5.2 Separating IO from Random Oracle Based Primitives

In this section we prove the following formalization of Theorem 5.1.1.

Theorem 5.2.1 (Theorem 5.1.1 formalized). *If* $\mathbf{NP} \not\subseteq \mathbf{co\text{-}NP}$ *then there is no fully black-box construction of IO from any primitive P that exists relative to a random oracle in a*

[3] Since the probability distribution here is over in nite-size oracles, we cannot assign probabilities to arbitrary events, but we can alternatively work with measurable sets.

[4] Here ϵ refers to the correctness error, and δ refers to the statistical closeness (see De nition 2.5.5).

[5] Here κ, the security parameter, is equal to the circuit size.

black-box way. This includes exponentially secure one-way functions and collision-resistant hash functions.

To prove Theorem 5.2.1 we will first prove a useful lemma (see Lemma 5.2.2) which, roughly speaking, asserts that for any pair of circuits C_1, C_2, either an attacker can guess which one is obfuscated in the random oracle model with a probability close to one, or that there is a way to obfuscate them into the same output circuit B. The latter could be used as a witness that C_1 and C_2 compute the same function, assuming that the obfuscation is an IO.

Lemma 5.2.2 (Distinguish or Witness)**.** *Let* iO *be an oracle aided randomized polynomial-time algorithm taking circuits as input such that for every length-preserving oracle f and every randomness r it holds that* $\mathsf{iO}^f_r(C) \equiv C$ *(i.e.,* iO *always outputs circuits with the same input/output functionality as the input circuit C). Then, at least one of the following holds:*

1. *There is an* inf*inite sequence of circuits* $(C^1_0, C^1_1), \ldots, (C^i_0, C^i_1), \ldots$ *such that* $|C^i_0| = |C^i_1|$ *for all i, and there exists a (computationally unbounded) $poly(\kappa)$-query A such that the following holds for all i:*

$$\Pr_{r,s,f,b}[A^f_s(B) = b : b \xleftarrow{\$} \{0,1\}, B = \mathsf{iO}^f_r(C^i_b)] \leq 1 - 1/\kappa^2$$

   where κ is the bit size of the circuits: $|C^i_0| = |C^i_1| = \kappa$.

2. $\mathbf{NP} = \mathbf{co\text{-}NP}$.

We will first prove Theorem 5.2.1 using Lemma 5.2.2, and then we will prove Lemma 5.2.2.

Proof of Theorem 5.2.1. In what follows we will always assume $\mathbf{NP} \neq \mathbf{co\text{-}NP}$. We will describe the proof for one-way functions, but it can be verified that the same proof holds for any primitive that holds relative to random oracles in a black-box way (see Definition 3.1.4).

Suppose (iO, S) is a fully black-box construction of IO from one-way functions. We use a random oracle f to implement the one-way function required by iO. By Lemma 5.2.2 and the assumption that $\mathbf{NP} \neq \mathbf{co\text{-}NP}$ we know that there is a computationally unbounded attacker A and an infinite sequence of equivalent and same-size circuits $(C^1_0, C^1_1), \ldots, (C^i_0, C^i_1), \ldots$ such that A breaks the security of IO over the challenge circuits (C^i_0, C^i_1) of length $|C^i_0| = \kappa = |C^i_1|$ by guessing which one of them is being obfuscated with probability $1 - 1/\kappa^2$. Let $\epsilon = 1/4$. By an averaging argument, with probability at least $1 - O(1/\kappa^2)$ over the choice of oracle f, it holds that the probability that A correctly guesses which one of (C^i_0, C^i_1) is being obfuscated is at least $1/2 + \epsilon$. Since the summation $\sum_i 1/i^2 = O(1)$ converges, by Borel-Cantelli lemma, for measure one of the random oracles f it holds that A ϵ-breaks the implemented obfuscator iO^f.

Now that A is a "legal" adversary, by definition of fully black-box IO, the security reduction $S^{f,A}$ shall break the one-way property of f. Algorithms A and S are both poly(κ)-query

attackers, and so the combination $B = S^A$ also asks only a polynomial number of queries to f and succeeds in breaking the one-wayness of f for nonzero measure of samples for f.

The existence of such B, however, is impossible since a random oracle f, with measure one, is secure against attackers who ask only a polynomial number of queries [IR89, GT00].[6]

□

Now to prove Lemma 5.2.2, we use the following lemma from [MP12].

Lemma 5.2.3 ([MP12]). *Suppose S is an oracle-aided PPT algorithm that calls oracle f and takes private input b 2 f0, 1g, randomness r, and common input z 2 f0, 1g (where κ is the security parameter) and outputs $c = S_r^f(z, b)$. For any $\delta = \delta(\kappa)$ 1/100, there is a (computationally unbounded) oracle-aided algorithm R such that for all z 2 f0, 1g at least one of the following holds.*

1. *If f is the random oracle, $R^f(z, c)$ asks* poly(κ/δ) *queries and correctly guesses the random bit b that $S_r^f(z, b)$ used to generate c with probability 1 $\delta(\kappa)$. Namely:*

$$\Pr_{f;r;b}[R^f(z, c) = b : b \xleftarrow{\$} f0, 1g, c = S_r^f(z, b)] \quad 1 \quad \delta(\kappa).$$

2. *There is a partial oracle f' of size* poly(κ) *and two random seeds r_0, r_1 and a message c such that $S_{r_0}^{f'}(z, 0) = c = S_{r_1}^{f'}(z, 1)$. In other words, there is a message c that could be opened into both $b = 0$ and $b = 1$ using random seeds r_0, r_1, and the queries asked by S during these two possible executions are all described by the partial function f'.*

Proof Sketch of Lemma 5.2.3. Let ϵ be a parameter to be chosen later. Let R be an attacker who maintains a list of \learned" oracle queries L and, given c sent by the sender for $b \xleftarrow{\$} f0, 1g$ and common input z, it adaptively asks the lexicographically first oracle query $x \notin L$ that has at least ϵ chance of being asked by sender S conditioned on the knowledge of (L, z). After asking such x from f, A adds $(x, f(x))$ to L. As long as such query x exists, R asks them. It was shown in [BMG07] that this learning algorithm asks, on average, at most m/ϵ number of queries where m = poly($|z|$) is the number of queries asked by the sender. So as long as ϵ = poly(κ/δ) this learning algorithm is efficient.

Now, let L be the final learned set by R. If conditioned on L it holds that both of $b = 0$ and $b = 1$ have at least ρ probability of being used by S, then by conditioning on the distribution of the sender's view on $b = 0$ or $b = 1$ all the unlearned queries remain *at most* $\epsilon/\rho = \sigma$-heavy. Now it is easy to see that if we sample a random view for S conditioned on $L, b = 0$ and $L, b = 1$ and call them V_0 and V_1, the probability that queries of V_0 and V_1 collide out of L is at most $m \quad \sigma \quad m \quad \epsilon/\rho$. For $\rho > m \quad \epsilon$ this probability is less than one, which means that if $\rho > m \quad \epsilon$, then there exists a consistent pair of views for S that he can use to output c for both cases of $b = 0$ and $b = 1$. This means that Case 2 happens.

[6] The works of [IR89, GT00] work with polynomial time Turing machines or circuits, however their goal is to fix the random oracle f before enumerating the attackers. However, if the attacker is fixed before the sampling of f, the proofs of [IR89, GT00] imply the one-wayness of f with measure one even if the fixed attacker is computationally unbounded.

Now let us assume that Case 2 does not happen. It means that for all executions of the algorithm A, when A is done with learning the ϵ heavy queries, the probability of either b = 1 or b = 0 conditioned on L is at most ϵ. This means that A can guess b correctly with probability 1 ϵ.

If we can choose ρ = $O(m/\epsilon)$ and ϵ = δ in the argument above (assuming that Case 2 does not happen) we get an attacker A that asks $O(m \epsilon/\epsilon) = O(m)$ queries. We can alternatively choose smaller ρ and cut A's execution after it asks $O(m/\delta)$ number of queries and use $\epsilon = \delta/10$. By an application of the Markov inequality A will ask more than $100(m/\delta)$ number of queries with probability at most ϵ, and so A will ask at most $O(m/\delta)$ number of queries and will guess b correctly with probability at least $2\epsilon < \delta$.

□

Remark 5.2.4. Mahmoody and Pass [MP12] proved a more general lemma ruling out (even \somewhere binding") non-interactive commitment schemes in the random oracle model. Lemma 5.2.3 is a special case of their result which su ces for our use. In the setting of [MP12] the security parameter is given to the parties in the form of 1 , but their proof handles parties who in addition receive an auxiliary z 2 $f0,1g$ and the parties' behavior could also depend on the given z.

Proof of Lemma 5.2.2. Consider the set of circuit pairs that are equivalent and of the same size: $C = f(C_0, C_1)\ j\ C_0 \quad C_1 \wedge jC_0j = jC_1jg$. We apply Lemma 5.2.3 for $\delta = 1/\kappa^2$ as follows. Use $(C_0, C_1) = z$ 2 C as the common input given to both parties. Let S be a sender strategy that, given input bit b, obfuscates C_b and sends out the obfuscated circuit B.

By Lemma 5.2.3 for each $(C_0, C_1) = z$ 2 C either of the following holds:

1. $A^f((C_0, C_1), B)$ can guess the random b in the random oracle model correctly with probability at least 1 $1/\kappa^2$.

2. There is a partial oracle f' of polynomial size and two random strings r_0, r_1 such that $iO^{f'}_{r_b}(C_b) = B$ for both b 2 $f0,1g$.

Note that if C_0 6 C_1 then Case (2) cannot happen as no such (f', r_0, r_1) can exist by perfect completeness of iO. Therefore, if Case (2) happens, the existence of (f', r_0, r_1) serves as an e ciently veri able proof that $C_0 \quad C_1$.

Now let C_a be the subset of C for which Case (1) holds. There are two cases:

1. C_a is not nite, in which case we have shown that Case 1 of the lemma holds.

2. If C_a is nite, then for all (except a nite number) of (C_0, C_1) 2 C we can e ciently prove that C_0, C_1 are equivalent circuits. This would give a proof system for proving the equivalency of two given circuits, but this problem is $\mathbf{co\text{-}NP}$-complete. Thus, $\mathbf{co\text{-}NP} = \mathbf{NP}$.

□

5.3 Hardness of Semi-Black-Box Constructions of IO

In this section, we prove Theorem 5.1.2. We will first show that approximate IO is still powerful enough to base public-key cryptography on private-key cryptography.

Theorem 5.3.1. *The existence of* $(1/6)$*-approximate IO and any one-way functions imply the existence of semantically secure public-key encryption schemes.*

In order to prove the above we will need the following theorem which is a more formal version of Theorem 5.1.3 and a direct consequence of the techniques used in Chapter 4 and in particular Theorems 4.4.18, 4.4.30, and 4.5.3.

Theorem 5.3.2. *Suppose* iO' *is an approximately correct obfuscation algorithm with error at most* ϵ' *in idealized model* I *where* I *is random trapdoor permutation oracle or the degree-*$O(1)$ *graded encoding model for nite rings. Suppose* $\epsilon'' \quad 1/\mathrm{poly}(\kappa)$*. Then there is another obfuscation algorithm* iO *in the plain model such that:*

> *The running time of* iO *is* $\mathrm{poly}(\kappa/\epsilon''(\kappa))$ *where* κ *is the size of the input circuit and it is approximately correct with error at most* $\epsilon = \epsilon' + \epsilon''$*.*
>
> *There is a simulator* Sim *in the idealized model that runs in time* $\mathrm{poly}(\kappa/\epsilon''(\kappa))$*, and for any circuit* C*, the distributions* $\mathsf{Sim}^I(\mathsf{iO}'^I(C))$ *and* $\mathsf{iO}(C)$ *have statistical distance* $\mathrm{negl}(|C|)$*.*

We first prove Theorem 5.1.2 using Theorems 5.3.2 and 5.3.1 then later prove Theorem 5.3.1.

Proof of Theorem 5.1.2 using Theorems 5.3.2 and 5.3.1 . Let P be any such primitive with implementation P relative to the idealized model I, and suppose iO is any such semi-black-box construction of IO from P. By Lemma 3.1.8 (the Composition Lemma), we conclude that $\mathsf{iO}' = \mathsf{iO}^P$ is a construction of IO in the idealized model I. This, together with Theorem 5.3.2 imply that there is a $(1/6)$-approximate IO in the plain model. Finally, by Theorem 5.3.1 and the existence of $(1/6)$-approximate IO implies that we can construct semantically secure public-key encryption from one-way functions. □

Proving Theorem 5.3.1 We now prove that $(1/6)$-approximate IO and one-way functions imply semantically secure public-key encryption. Therefore, any provably secure construction of $(1/6)$-approximate IO would enable us to take any one-way functions and construct a secure public-key encryption scheme from it. In the terminology of [IR89] it means that Cryptomania collapses to Minicrypt if $(1/6)$-approximate IO exists.

Intuition. Sahai and Waters [SW14] showed that IO and OWF imply PKE. Here we show that the very same construction, when instantiated using *approximate* IO, leads to \approximately correct" and \approximately secure" public-key encryption. Then, using a result of [Hol06] we amplify the soundness and correctness to get a full edged semantically secure public key encryption scheme.

Definition 5.3.3 (Approximate correctness and security for PKE)**.** We call a public-key bit-encryption scheme (Gen, Enc, Dec) for message space $f0, 1g$ $\epsilon(\kappa)$-correct if

$$\Pr[\mathsf{Dec}_{dk}(\mathsf{Enc}_{ek}(b)) = b\colon (ek, dk) \quad \mathsf{Gen}(1^{\kappa}), b \overset{\$}{\leftarrow} f0, 1g] \quad 1 \quad \epsilon(\kappa)$$

where the probability is over the randomness of the key generation, encryption, decryption, and the bit b. We call (Gen, Enc, Dec) $\delta(\kappa)$-secure if for any PPT adversary A, it holds that

$$\Pr[A(pk, \mathsf{Enc}_{pk}(b)) = b] \quad 1/2 + \delta(\kappa)$$

where the probability is over the randomness of generation, encryption, the adversary, and bit b.

Holenstein [Hol06] shows how to amplify any ϵ-correct and ϵ-secure PKE into a full edged (semantically secure) PKE for su ciently small ϵ.

Theorem 5.3.4 (Implied by Corollary 7.8 in [Hol06])**.** *Suppose* (Gen, Enc, Dec) *is ϵ-correct and δ-secure for* constants ϵ, δ *such that* $(1 \quad 2\epsilon)^2 > 2\delta$*. Then there exists a semantically secure PKE.*

Theorem 5.3.5 below asserts that approximate IO and one-way functions imply approximately correct and approximately secure PKE.

Theorem 5.3.5 (Approximate IO + OWF $)$ Approximate PKE)**.** *If ϵ-approximate IO and one-way functions exist, then there is an ϵ-correct and* (ϵ + negl(κ))*-secure public-key bit encryption scheme.*

We rst prove Theorem 5.3.1 using Theorem 5.3.5 and then will prove 5.3.5.

Proof of Theorem 5.3.1. Because $(1 \quad 2 \quad 1/6)^2 > 2 \quad 1/6$, Theorem 5.3.1 follows immediately from Theorem 5.3.4 and the following Theorem 5.3.5 using $\epsilon = 1/6$. □

In the rest of this section we prove Theorem 5.3.5.

Proof of Theorem 5.3.5. We show that the very same construction of PKE from IO and OWF presented by Sahai and Waters [SW14], when instantiated with an ϵ-approximate IO, has the demanded properties of Theorem 5.3.5.

Properties of the construction of [SW14]. We rst describe the abstract properties of the construction of [SW14] (for PKE using IO and OWFs) and its security proof that we need to know.

Construction/correctness:

1. The key generation process generates a circuit C and publishes $\mathsf{iO}(C) = B$ as public key where iO is an IO scheme.

2. The encryption simply runs B on (r, b) where r is the encryption randomness and b is the bit to be encrypted.
3. The scheme has completeness 1.

Security: [SW14] proves the security of the construction above by showing that no PPT algorithm can distinguish between the following two random variables X_0, X_1 de ned as:

- X_b $(\mathsf{iO}_s(C), C(r, b))$ $(B, C(r, b))$ where s is the randomness for iO. When clear from the context we drop the randomness s and simply write $\mathsf{iO}(C)$ denoting it as a random variable over the randomness of iO.

It can be veri ed by inspection that the proof of [SW14] for indistinguishability of X_0 and X_1 does *not* rely on completeness of the obfuscation iO and only relies on its indistinguishability (when applied to circuits with the same functions). We will rely on this feature of the proof of [SW14] in our analysis.

Below we analyze the correctness and security of the construction of [SW14] when iO is an ϵ-approximate IO.

Correctness. By the de nition of ϵ-approximate IO and ϵ-correct bit encryption, and the fact that the [SW14] construction has perfect completeness when iO is an IO, it follows that the completeness of the new scheme (when b is also chosen at random) is at least $1 - \epsilon$. Thus the scheme is ϵ-correct.

Security. First recall that for the basic construction of [SW14] using (perfect) IO, no PPT attacker A can guess b with probability better than $1/2 + \mathsf{negl}(\kappa)$ when b is chosen at random and A is given a sample from the random variable X_b (for random b). As we mentioned above, the proof of this statement does not rely on the correctness of the used IO and only relies on its indistinguishability.

Now we want to bound the distinguishing advantage of PPT adversaries between the following random variables Y_0, Y_1:

Y_b $(B, B(r, b))$ $(\mathsf{iO}_s(C), \mathsf{iO}_s(C)(r, b))$ where s is the randomness of the obfuscator iO.

The di erence between Y_b's and X_b's stems from the fact that the public-key $B = \mathsf{iO}(C)$ no longer computes the same exact function as the circuit C as the obfuscation only guarantees *approximate* correctness. We reduce the analysis of the new scheme to the original analysis of [SW14].

By the analysis of [SW14] we already know that if any PPT A is given a sample from X_b for a random b it has at most $1/2 + \mathsf{negl}(\kappa)$ chance of correctly guessing b. Also note that the distributions X_b and Y_b for a *random* b are ϵ-close due to the ϵ-correctness of the obfuscation. More formally, the distributions X_b and Y_b could be de ned over the same sampling space

using: random seeds of key generation, obfuscation, encryption, and bit b. This way with probability 1 ϵ (and by the ϵ-approximate correctness of the obfuscation) the actual sampled values of X_b and Y_b will be equal, and this implies that they are ϵ-close. As a result, when we switch the distribution of the challenge given to the adversary and give a sample of Y_b (for random b) instead of a sample from X_b, the adversary's chance of guessing b correctly can increase at most by ϵ and reach at most 1/2 + negl + ϵ. Therefore, the new scheme is (ϵ + negl)-secure according to De nition 5.3.3.

□

5.4 Separating IO from TDP and Constant-Degree GEM

In this section we improve upon the previous section's result by showing that building IO in the examined idealized models is not just as hard as basing public-key encryption on private-key encryption but in fact impossible to achieve in a fully black-box way.

A fully-black-box construction of approximate computational CIO from primitive *P* could be de ned through a combination of De nitions 3.1.2 and 2.5.5. Here we emphasize that the input circuits do not have any oracle gates while the obfuscation algorithm and the nal circuits could use the oracle implementing *P*. This seemingly restricted model is in fact su cient for all known applications.

We are now ready to formally prove this section's result. First we formalize the statement by specifying the way *P* is constructed in the idealized models.

Theorem 5.4.1 (Main Result). *Assuming the existence of one-way functions and* $\mathbf{NP}$ *6* $\mathbf{coAM}$*, there is no fully-black-box construction of IO from any primitive P that has a oracle- xed black-box construction (see De nition 3.1.4) in the random trapdoor permutation oracle or the degree-*$O(1)$ *graded encoding model for any nite ring.*

In fact, we prove a stronger separation that holds for approximate computational CIO as well.

Theorem 5.4.2. *Assuming there is no* (ϵ, δ)*-approximate statistical CIO, there is no fully-black-box construction of* (ϵ', δ')*-approximate computational CIO for any* ϵ' ϵ κ $^{(1)}, \delta'$ δ (1) *from any of the primitives that can be constructed in the idealized models listed in Theorem 5.4.1.*

In order to prove this we will rely on the result of a previous work due to Brakerski et al. [BBF16], which states that (approximate) statistically-secure IO is impossible to achieve.

Theorem 5.4.3 (Impossibility of Approximate Statistically-secure IO [BBF16]). *Suppose one-way functions exist,* $\mathbf{NP}$ *6* $\mathbf{coAM}$*, and* δ, ϵ: N 7! [0, 1] *are such that* $2\epsilon(\kappa) + 3\delta(\kappa) < 1$ $1/\text{poly}(\kappa)$*. Then there is no* (ϵ, δ)*-approximate statistically-secure CIO (see De nition 2.5.4) for all poly-size circuits.*

Proving Theorem 5.4.1 using Theorems 5.4.3 and 5.4.2. Theorem 5.4.3 rules out (ϵ, δ)-approximate statistical CIO (assuming OWFs and $\mathbf{NP}$ *6* $\mathbf{coAM}$) for some ϵ = $1/\text{poly}(\kappa)$ and δ = 0.3. Thus, if we choose ϵ^0 = $\epsilon/2$ and δ^0 = $\delta/2$, then Theorem 5.4.1 follows from Theorems 5.4.2 and 5.4.3.

In the following we will focus on proving Theorem 5.4.2.

Remark 5.4.4 (The need for constant δ.)**.** Our proof of Theorem 5.4.2 crucially relies on the fact that δ δ^0 (1) which in turn requires δ (1). Thus, the separation holds because the attacker of [BBF16] could achieve δ 1/3 (as opposed to just $1/\text{poly}(\kappa)$). More technically, our proof will make use of Lemma 2.2.2 rather than the Borel-Cantelli lemma, and that is the source of our need for δ (1). However, in case one can improve the result of [BBF16] to cover the setting of ϵ = $1/\text{poly}(\kappa)$ and δ = 1 α for arbitrary small α = $1/\text{poly}(\kappa)$, then our Theorem 5.4.2 could be improved to any δ^0 = δ $1/\text{poly}(\kappa)$. In fact the proof will be simple and will not use our Lemma 2.2.2 and could be based on the Borel-Cantelli lemma (see the end of this section for a sketch).

Remark 5.4.5 (Ruling out relativizing constructions)**.** In Theorem 5.4.1 we focus on ruling out fully-black-box constructions. However, the proof can be extended to rule out relativizing constructions (of IO from the set of listed primitives) using standard techniques and the fact that an optimal statistical distinguisher can be implemented in $\mathbf{PSPACE}$. In particular, the separating oracle would be a random sample from the idealized oracle I *I* and an oracle for a $\mathbf{PSPACE}$-complete oracle. However, interestingly, in our case the sampled I *I* would only work with *constant* measure (which is enough since it is still a positive measure) due to using Lemma 2.2.2 as opposed to measure one, which is typically the case in black-box separations.

Proof of Theorem 5.4.2. In the following, let *Q* denote the primitive of (ϵ^0, δ^0)-approximate computational CIO. Also let *P* be any primitive that can be constructed in the idealized models listed in Theorem 5.4.1 (according to De nition 3.1.4), and let P be the implementation of *P* relative to *I*.

For sake of contradiction, in the following we let Q be the fully-black-box construction of *Q* from *P*. We will rst make use of Lemma 3.1.8 (the Composition Lemma) to show that *Q* could also be implemented relative to *I* as well. Then we rule out the existence of black-box constructions of *Q* from *I* to conclude that Q could not exist.

In the following we will use Theorems 5.3.2 and 5.4.3 to rule out the possibility of any *oracle- xed black-box* construction of *Q* relative to *I* which (with Lemma 3.1.8) shows that Q could not exist.

Let ϵ^{00} = ϵ ϵ^0 $1/\text{poly}(\kappa)$ and δ^{00} = δ δ^0 (1). Since P is a construction of *P* relative to *I*, we have that iO^{0I} = $(Q^P)^I$ is an ϵ^0-approximate obfuscation mechanism relative to *I*. Let iO be the ϵ-approximate obfuscator in the plain model that exists due to Theorem 5.3.2. The assumption in Theorem 5.4.2 is that iO cannot be an (ϵ, δ)-approximate statistical CIO. Therefore, there exists a computationally unbounded adversary A and an in nite sequence of circuit pairs $(C_0^1, C_1^1), \ldots, (C_0^i, C_1^i), \ldots$ such that for all i: $|C_0^i| = |C_1^i|$, C_0^i C_1^i, and $\Pr_{b \in \{0,1\}}[A(\text{iO}(C_b^i)) = b]$ $1/2 + \delta(\kappa)/2$.

Now consider another attacker A' in the idealized model $\mathcal{I}$ which, given a circuit B' as input, runs the simulator of Theorem 5.3.2 to get the circuit $B = \mathsf{Sim}^{\mathcal{I}}(B')$ and then runs A over B to output whatever A does. By the property of the simulator Sim we conclude that A' is an e cient query (computationally unbounded) attacker in the idealized model $\mathcal{I}$ that achieves

$$\Pr_{b \leftarrow \{0,1\},\mathcal{I}}[A'^{\mathcal{I}}(\mathsf{iO}'^{\mathcal{I}}(C_b^i)) = b] \geq 1/2 + \delta(\kappa)/2 \geq \mathsf{negl}(\kappa)$$

where $|C_0^i| = |C_1^i| = \kappa$.

A crucial point is that the above probability is *also over the randomness of the oracle* $\mathcal{I}$ for every i, while we are interested in *xing* $\mathcal{I}$ and getting a successful attack for in nitely many pairs of circuits at the same time. By a simple averaging argument we can get:

$$\Pr_{\mathcal{I}}\left[\Pr_{b \leftarrow \{0,1\}}[A'^{\mathcal{I}}(\mathsf{iO}'^{\mathcal{I}}(C_b^i)) = b] \geq 1/2 + \delta'(\kappa)/2\right] \geq \delta''(\kappa)/2 \geq \mathsf{negl}(\kappa).$$

Thus, if we de ne the event E_i over the sampled oracle $\mathcal{I}$ as:

$$E_i \text{ holds if: } \Pr_{b \leftarrow \{0,1\}}[A'^{\mathcal{I}}(\mathsf{iO}'^{\mathcal{I}}(C_b^i)) = b] \geq 1/2 + \delta'/2$$

then we get $\Pr[E_i] \geq \delta''(\kappa)/2 - \mathsf{negl}(\kappa) \geq \delta''/3$ for every $i \in \mathbb{N}$. Now we can apply Lemma 2.2.2 to conclude that, with probability at least $\delta''/3$ over the choice of $\mathcal{I}$, an in nite number of the events E_i's would happen at the same time for $\mathcal{I}$. We call $\mathcal{I}$ a good oracle if it is indeed the case that in nitely many of the events E_i's happen over $\mathcal{I}$. By de nition, for any good oracle $\mathcal{I}$, the attacker A' successfully breaks $(Q^P)^{\mathcal{I}}$ (as an implementation of Q in model $\mathcal{I}$) over in nitely many pairs of circuits while asking only an e cient number of oracle queries to $\mathcal{I}$. The existence of such A' who breaks $(Q^P)^{\mathcal{I}}$ for non-zero (in fact $\delta''/3$) measure of the choice of the oracles $\mathcal{I}$ prevents Q^P from being a oracle- xed black-box construction of Q relative to $\mathcal{I}$. □

Case of $\delta' \leq 1 - 1/\mathrm{poly}(\kappa)$. Theorem 5.4.2 was su cient for us to derive Theorem 5.4.1, however that is not the strongest separation one can imagine for approximate computational CIO as it does not cover the case of $1 - 1/\mathrm{poly}(\kappa)$. The work of [BBF16] shows that whenever $2\epsilon + \delta > 1$ then there is in fact a way to achieve (ϵ, δ)-approximate statistical CIO. Thus one can imagine the possibility that the result of [BBF16] could ultimately be improved to rule out (ϵ, δ)-approximate statistical CIO for $O(\epsilon) + \delta < 1 - 1/\mathrm{poly}(\kappa)$. Below, we show that such a result, if proved, could be used to derive lower bounds on the complexity of (ϵ', δ')-approximate computational CIO for $\delta' \leq 1 - 1/\mathrm{poly}(\kappa)$.

Theorem 5.4.6. *If there is no (ϵ, δ)-approximate statistical CIO for $\delta = 1 - \rho$ for su ciently small $\rho = 1/\mathrm{poly}(\kappa)$ (e.g., $\rho = 1/\kappa^4$ su ces), then there is no fully-black-box construction of $(\epsilon', \delta' = 1 - \sqrt{\rho})$-approximate computational CIO for any $\epsilon' \leq \epsilon - \kappa^{-\omega(1)}$ from the primitives of Theorem 5.4.1.*

Thus, the main difference between Theorem 5.4.2 and Theorem 5.4.6 is that in Theorem 5.4.6 we cover the case of $\delta' = 1 - 1/\text{poly}(\kappa)$, but we also rely on stronger assumption that $\delta = 1 - 1/\text{poly}(\kappa)$.

Proof of Theorem 5.4.6. The proof is identical to that of Theorem 5.4.2 except for the following. Since the attackers A and A' will succeed in guessing the correct circuit with probability $1 - 1/\text{poly}(\kappa) - 1$ we can do a better averaging argument to get a better attack after fixing the oracle. Namely, define the event E_i as:

$$E_i \text{ holds if: } \Pr_{b \in \{0,1\}}[A'^{O'}(iO'^{O'}(C_b^i)) = b] \geq 1 - \sqrt{\rho(\kappa)/2}$$

where κ is the size of the circuits C_0^i, C_1^i. Then we can conclude that $\Pr[E_i] \geq 1 - 10\sqrt{\rho(\kappa)}$. Now, since the events E_i happen with large probability and that $\sum 10\sqrt{\rho(\kappa)} < 1$ we can apply the Borel-Cantelli lemma (Lemma 2.2.1) to conclude that with measure *one* over the choice of the oracle O all but finitely many of E_i's would happen. The rest of the proof remains unchanged. □

Part II

Monolithic Separations for Indistinguishability Obfuscation

Chapter 6

Extending the Black-box Framework

6.1 Introduction

Before we delve into separating IO from more sophisticated primitives we will have to address the current black-box framework since it is incapable of capturing constructions of IO from more advanced primitives. We will start be motivating the need for this new framework.

Recall that the black-box framework (discussed in Chapter 3) restricts to constructions that use the primitive and the adversary (in the security reduction) as a black-box. The reason that this framework does not work out-of-the-box for our separation results is that the constructions of IO from powerful encryption primitive allow for a very natural *non*-black-box use. In fact, the construction of IO from compact functional encryption (FE) [AJ15, BV15, AJS17] is non-black-box in its use of functional encryption. This is not a coincidence (or, just one example) and many applications of functional encryption (as well as other powerful encryption schemes) and IO are non-black-box [Gen09, SW14, BZ14, GPS16, GPSZ17]. Note that the di erence between these powerful primitives and the likes of one-way functions, hash functions, etc., is that these powerful primitives include subroutines that take arbitrary circuits as inputs. Therefore, it is very easy to *self-feed* the primitive. In other words, it is easy to plant gates (or function calls) of its own subroutines (or, subroutines of other cryptographic primitives) inside such a circuit that is then fed to it as input. For example, the construction of IO from FE plants FE's encryption subroutine as a gate inside the circuit for which it issues decryption keys. This makes FE a \special" primitive in that at least one of its subroutines takes an arbitrary circuit as input and we could plant code of its subroutines in this circuit. Consequently, the obtained construction would be non-black-box in the underlying primitive. This special aspect is present in *all* of the primitives that we aim to separate IO from in the subsequent chapters. For example, one of the subroutines of predicate encryption (see De nition 2.6.4) takes a circuit as input and this input circuit is used to test whether the plaintext is revealed during the decryption or not. Along similar lines, evaluation subroutine of a fully-homomorphic scheme (see De nition 2.6.6) is allowed to take as input a circuit that is executed on an encrypted message.

The above \special" aspects of the encryption functionalities (i.e. that they take as

input general circuits or Turing machines and execute them) is the main reason that many of the applications of these primitives are non-black-box constructions. Therefore, any e ort to prove a meaningful impossibility result should aim for proving the result with respect to a more general framework than that of [IR89, RTV04]. In particular, this more general framework should incorporate the aforementioned non-black-box techniques as part of the framework itself.

The previous works of Brakerski, Katz, Segev, and Yerukhimovich [BKSY11] and the more recent works of Asharov and Segev [AS15, AS16] are very relevant to our studies here. All of these works also deal with proving limitations for primitives that in this work we call special (i.e. those that take general circuits as input), and prove impossibility results against constructions that use these special primitives while allowing some form of oracle gates to be present in the input circuits. A crucial point, however, is that these works still put some limitation on *what* oracle gates are allowed, and some of the subroutines are excluded. The work of [BKSY11] proved that the primitive of Witness Indistinguishable (WI) proofs for $\mathbf{NP}^O$ statements where O is a random oracle does not imply key-agreement protocols in a black-box way. However, the WI subroutines themselves are not allowed inside input circuits. The more recent works of [AS15, AS16] showed that by using IO over circuits that are *allowed* to have one-way functions gates one cannot obtain collision resistant hash functions or (certain classes of) one-way permutations families (in a black-box way). However, not all of the subroutines of the primitive itself are allowed to be planted as gates inside the input circuits (e.g., the evaluation procedure of the IO).

As a result, we are going to revisit the models used in [BKSY11, AS15, AS16] who allowed the use of one-way function gates inside the given circuits and study a model where there is no limitation on what type of oracle gates could be used in the circuits given as input to the special subroutines, and in particular, the primitive's own subroutines could be planted as gates in the input circuits. We believe a model that captures the \gate plantation" technique without putting any limitation on the types of gates used is worth to be studied directly and at an abstract level, due to actual *positive* results that exactly bene t from this \self-feeding" non-black-box technique. For this goal, here we initiate a formal study of a model that we call the *monolithic* model, which captures the above-described non-black-box technique that is commonplace in constructions that use primitives with subroutines that take arbitrary circuits as input.

More formally, suppose *P* is a primitive that is special as described above, namely, at least one of its subroutines might receive a circuit or a Turing machine C as input and executes C internally in order to obtain the answer to one of its subroutines. Examples of *P* are predicate encryption, fully homomorphic encryption, etc. A *monolithic* construction of another primitive *Q* (e.g., IO) from *P* will be allowed to plant the subroutines of *P* inside the circuit C as gates with no further limitations. To be precise, C will be allowed to have oracle gates that call *P* itself. Some of major examples of *non-black-box* constructions that fall into this monolithic model are as follows.

> Gentry's bootstrapping construction [Gen09] plants FHE's own decryption gates inside a circuit that is given as input to the evaluation subroutine. This trick falls into the

monolithic framework since planting gates inside evaluation circuits is allowed.

The bootstrapping of IO for $\mathbf{NC}_1$ (along with FHE) to obtain IO for $\mathbf{P}/\mathbf{poly}$ [GGH+13b]. This construction uses P that includes both IO for $\mathbf{NC}_1$ and FHE, and it plants the FHE decryption gates inside the $\mathbf{NC}_1$ circuit that is obfuscated using IO for $\mathbf{NC}_1$. Analogously, bootstrapping methods using one-way functions [App14, CLTV15] also fall in our framework.

The construction of IO from functional encryption [AJ15, BV15, AJS17] plants the functional encryption scheme's encryption subroutine inside the circuits for which decryption keys are issued. Again, such a non-black-box technique does fall into our monolithic framework. We note that the constructions of obfuscation based on constant degree graded encodings [Lin16] also fit in our framework.

The above examples show the power of the monolithic model in capturing one of the most commonly used non-black-box techniques in cryptography and especially in the context of powerful encryption primitives.

What is *not* captured by the monolithic model? It is instructive to understand the kinds of non-black-box techniques not captured by our extension to the black-box model. This model does not capture non-black-box techniques that break the computation of a primitives sub-routines into smaller parts | namely, we do not include techniques that involve partial computation of a sub-routine, save the intermediate state and complete the computation later. In other words, the planted sub-routines gates must be executed in one-shot. Therefore, in our model given just an oracle that implements a one-way function it is *not* possible to obtain garbled circuits that evaluate circuits with one-way function gates planted in them. For example, Beaver's OT extension construction cannot be realized given just oracle access to a random function.

However, a slight workaround (though a bit cumbersome) can still be used to give meaningful impossibility results that use garbled circuits (or, randomized encodings more generally) in our model. Specifically, garbled circuits must now be modeled as a special primitive that allows for inputs that can be arbitrary circuits with OWF gates planted in them. With this change the one-way function gate planted inside circuit fed to the garbled circuit construction is treated as a individual unit. With this change we can realize Beaver's OT extension construction in our model.

In summary, intuitively, our model provides a way to capture \black-box" uses of the known non-black-box techniques. While the full power of non-black-box techniques in cryptography is yet to be understood, virtually every known use of non-black-box techniques follows essentially the same principles, i.e. by planting subroutines of one primitive as gates in a circuit that is fed as input to the same (or, another) primitive. Our model captures any such non-black box use of the considered primitives.

6.2 Our Results

In this chapter we will describe ideas that lead to an extension to the black-box framework of [IR89, RTV04], which we call the *monolithic framework* where a common *non-black-box* technique is allowed in the constructions.

The goal of this chapter is to explain and justify the way we de ne \monolithic" constructions that use speci c primitives such as witness encryption (see De nition 2.6.1). As it will become clear later, our de nition of monolithic constructions is meaningful with respect to a \special" primitive with a \special" subroutine. But, to be fully formal, we will give formal de nitions of what this monolithic model means when we use the exact set of primitives mentioned in our main theorem. This way, we can give *full formal de nitions as to what we mean by an monolithic construction for our speci c primitives* that are needed for the separations in Theorem 7.1.1, and we also give full proofs of separations for them under this more relaxed black-box notion. However, we emphasize that, for a new primitive P, one still has to give a formal de nition of what it means to use P in an monolithic way. What we discuss in this section lays down the main intuitive features that, if P possesses them, a monolithic use of P is possible.

Intuitively, this monolithic model allows the construction of a primitive Q based on another primitive P to plant oracle gates (or oracle calls) to P itself in the circuits (or Turing machines) in generic computations done by P whenever Q has the choice of doing so by choosing certain inputs given to P. For example, to get identi cation protocols from one-way functions and zero-knowledge proofs [FS87, FFS88] one has to use the code of one-way functions and feed it to the zero-knowledge protocol used. This makes the nal construction *non-black-box* according to the [RTV04] de nition, however we would like to de ne a general abstract extended model of black-box constructions (i.e. monolithic constructions) that allows such techniques.

In Section 6.4 we get into the details of how to de ne black-box reductions for a class of \special" primitives with a special subroutine as described above that fall under the monolithic framework. In this technical overview, however, for sake of simplicity and concreteness at the same time, we focus on developing a formal de nition of what it means to use *witness* encryption in a monolithic way.

6.3 A Concrete Definition for Case of WE

Consider the primitive of witness encryption (Enc, Dec) where one can run Enc(a, m) = c to encrypt a message m under a circuit a, and then later on, one can also try to decrypt c by using a \witness" w for which $a(w) = 1$.[1] A black-box construction (see De nition 3.1.2) of a primitive based on witness encryption is limited to only encrypting messages under circuits a that are in the *plain* model. That is because the input-output function de nition of primitive witness encryption (as stated above) only accepts plain circuits as input. However, in many applications of witness encryption, when we use witness encryption, we end up

[1] This way of de ning witness encryption is based on the $\mathbf{NP}$ complete problem of circuit satis ability.

using *oracle* circuits a^O. The reason that doing so is ne (even though the de nition of witness encryption requires plain circuits) that those procedures of O would be eventually implemented e ciently and could be turned into circuits (using the Cook-Levin reduction).

A monolithic construction from witness encryption is exactly allowed to do the above trick (of planting oracle gates in the input circuits) but it does so *regardless* of having an e cient implementation for the subroutines of WE. In other words, even if we use an ine cient oracle to implement WE, we still get an ine cient implementation for the constructed primitive. Now, we can let a to be an oracle circuit calling either of (Enc, Dec) freely, when (Enc, Dec) themselves are available as oracle subroutines (and might be *ine ciently* implemented). We emphasize that, by allowing this extension, we might eventually get multiple levels of *nested* oracle calls (since the circuit given to the decryption, can potentially have decryption gates itself), and allowing this to happen makes this model more powerful. However we should also be careful that this nested sequence of calls does not lead to exponential time.

Monolithic Construction = Fully Black-box Use of an Extended Primitive The above-mentioned example of how a monolithic construction using WE works shows that monolithic constructions using witness encryption could be interpreted in an equivalent way in which the construction is in fact *fully* black-box (see De nition 3.1.2), but it uses a *stronger* \extended" variant of the WE primitive where we allow oracle gates (calling the same primitive, namely witness encryption) to be planted in the circuits given as input to the encryption subroutine. Note that the extended WE is technically a *di erent* primitive than WE (because for ine cient implementations of WE, we would accept ine cient circuits as inputs as well), even though the two primitives are equivalent in a *non-black-box* way.

The following is the way we de ne extended WE in a \symmetric" way such that either the given input instance is a circuit (by interpreting it so and running it over the witness), or the instance is actually an input to a circuit and it is the *witness* that is the circuit. See De nition 6.5.9 for a more detailed version.

Definition 6.3.1 (Extended Witness Encryption)**.** Let V be an *oracle* algorithm that takes \instance" a and \witness" w as inputs, interprets a as an oracle-aided circuit, then outputs $a^{(\cdot)}(w)$. An *extended witness encryption* scheme for a relation given by V consists of two PPT algorithms $(\mathsf{Enc}, \mathsf{Dec}_V)$ de ned as follows:

> $\mathsf{Enc}(a, m, 1^\lambda)$: given an instance $a \in \{0,1\}^*$ and a message $m \in \{0,1\}^\lambda$ outputs $c \in \{0,1\}^*$.
>
> $\mathsf{Dec}_V(w, c)$: given ciphertext c and \witness" string w, it either outputs a message $m \in \{0,1\}^\lambda$ or $\perp$.

An extended witness encryption scheme satis es the the same completeness and security properties as the standard De nition of WE (see De nition 2.6.1).

We can also de ne a symmetric variant of extended WE where the relation V interprets w as the circuit instead and runs $w(a)$. In either case, we note that, since V is a relation that

acts as a universal circuit, the running time of V(w, a) is guaranteed to be poly(n) where $n = |a| + |w|$. That is due to the fact that each recursive call is invoked on inputs of a smaller size (in fact of length n 1) which results in a halting execution of time at most $O(n^2)$.

Using the same idea, one can de ne extended variants of many other primitives (see De nition 6.5.10). Finally, after de ning what an extended WE primitive is, we can formally de ne what it means to have an monolithic construction that uses WE:

Definition 6.3.2 (Monolithic Construction of WE)**.** Suppose Q is a primitive and $\widetilde{\mathsf{WE}}$ is an extended version of WE de ned as in De nition 6.3.1. Any fully black-box construction (see De nition 3.1.2) for Q from $\widetilde{\mathsf{WE}}$ (i.e. an extended version of WE) is also a *monolithic* construction of Q from WE.

Again, the idea extends beyond WE and we apply it to all the other primitives that we deal with in our main Theorem 7.1.1.

Monolithic Separations through Fully Black-Box Separations. The crucial point here is that because monolithic constructions are in fact fully black-box constructions based on the extended variants of the same primitive, at least we will have a path to prove impossibility results in this monolithic model via proving fully black-box constructions; however, the catch is that we have to do so with respect to a much stronger variant of the original primitive, namely its more powerful extensions.

6.3.1 A Transitivity Lemma for Deriving More Separations

The way we de ned monolithic constructions (based on black-box constructions using the extended primitive) allows us to prove a transitivity lemmas that will then pave our way towards proving more separations in the monolithic framework. Namely, it can be shown that if P, Q, R are cryptographic primitives and (1) there is an monolithic construction of the *extended* variant of Q from P and (2) there is an monolithic construction of R from Q, then there is a monolithic construction of R from P (see Lemma 6.6.3 and its proof). Therefore, to prove that a primitive P (e.g., predicate encryption) does not imply IO, we could employ the following argument outline:

1. Separate IO from WE in the monolithic model. In other words, prove that there is no fully black-box construction of IO from the extended variant of WE.

2. Prove that the *extended* variant of the primitive P could be obtained from witness encryption in a monolithic way.

We will apply the above idea to various forms of witness encryption and di erent all-or-nothing encryption primitives P.

6.4 An Abstract Extension of the Black-Box Model

In what follows, we will gradually develop an extended framework of constructions that includes the fully black-box framework of [RTV04] and allows certain non-black-box techniques by default. This model uses steps already taken in works of Brakerski, Katz, Segev, and Yerukhimovich [BKSY11] and the more recent works of Asharov and Segev [AS15, AS16] and takes them to the next level by allowing even non-black-box techniques involving `self-calls' [AJ15, BV15, AJS15]. In a nutshell, this framework applies to `special' primitives that accept generic circuits as input and run them on other inputs; therefore one can plant oracle gates to the same primitives inside those circuits. We will de ne such constructions using the fully black-box framework by rst extending these primitives and then allowing the extensions to be used in a black-box way.

Special Primitives Receiving Circuits as Input At a very high level, we call a primitive `special', if it takes circuits as input and run those circuits as part of the execution of its subroutines, but at the same time, the exact de nition depends on the execution of the input circuit only as a `black-box' while the exact representation of the input circuits do not matter. In that case one can imagine an input circuit with oracle gates as well. We will simply call such primitives special till we give formal de nitions that de ne those primitives as `families' of primitives indexed by an external universal algorithm.

Here is a list of examples of special primitives.

Zero-knowledge proofs of circuit satisfiability (ZK-Cir-SAT). A secure protocol for ZK-Cir-SAT is an interactive protocol between two parties, a prover and a veri er, who take as input a circuit C. Whether or not the prover can convince the veri er to accept the interaction depends on the existence of x such that $C(x) = 1$. This de nition of the functionality of ZK-Cir-SAT does not depend on the speci c implementation of C and only depends on executing C on x `as a black-box'.

Fully homomorphic encryption (FHE). FHE is a semantically secure public-key encryption where in addition we have an evaluation sub-routine Eval that takes as input a circuit f and ciphertexts $c_1, \ldots, c_k$ containing plaintexts $m_1, \ldots, m_k$, and it outputs a new ciphertext $c = \text{Eval}(f, c_1, \ldots, c_k)$ such that decrypting c leads to $f(m_1, \ldots, m_k)$. The correctness de nition of the primitive FHE only uses the input-output behavior of the circuit f, so FHE is a special primitive.

Encrypted functionalities. Primitives such as attribute, predicate, and functional encryption all involve running some generic computation at the decryption phase before deciding what to output. There are two ways that this generic computation could be fed as input to the system:

- Key policy [SW05, GPSW06]: Here the circuit C is given as input to the key generation algorithm and then $C(m)$ is computed over plaintext m during the decryption.

– Ciphertext policy [BSW07]: Here the circuit C is the actual plaintext and the input m to C is used when issuing the decryption keys.

Both of these approaches lead to special primitives. For example, for the case of predicate encryption, suppose we use a predicate veri cation algorithm P that takes (k, a), interprets k as circuit and runs $k(a)$ to accept or reject. Such P would give us the key policy predicate encryption. Another P algorithm would interpret a as a circuit and runs it on k, and this gives us the ciphertext policy predicate encryption. In other words, one can think of the circuit C equivalent to P(k,) (with k hard coded in it, and a left out as the input) being the \input" circuit given to the KGen subroutine, or alternatively one can think of P(, a) (with a hardcoded in it, and k left out as the input) to be the \input" circuit given to the Enc subroutine. In all cases, the correctness and security de nitions of these primitives only depend on the input-output behavior of the given circuits.

Witness Encryption. The reason that witness encryption is a special primitive is very similar to the reason described above for the case of encrypted functionalities. Again we can think of V(, a) as the circuit given to the Enc algorithm. In this case, the de nition of witness encryption (and it security) only depend on the input-output behavior of these `input circuits' rather their speci c implementations.

Indistinguishability Obfuscation. An indistinguishability obfuscator takes as input a circuit C and outputs B that can be used later on the compute the same function as C does. The security of IO ensures that for any two di erent equally-sized and functionally equivalent circuits C_0, C_1, it is hard to distinguish between obfuscation of C_0 and those of C_1. Therefore, the correctness and security de nitions of IO depend solely on the input-output behavior (and the sizes) of the input circuits.

When a primitive is special, one can talk about \extensions" of the same primitive in which the circuits that are given as input could have oracle gates (because the primitive is special and so the de nition of the primitive still extends to such inputs).

6.5 An Abstract Model for Extended Primitives and Constructions

We de ne special primitives as `restrictions' of (a family of) primitives indexed by a subroutine W to the case that W is a universal circuit evaluator. We then de ne the extended version to be the case that W accepts oracle-aided circuits. More formally we start by de ning primitives indexed by a class of functions.

Definition 6.5.1 (Indexed primitives)**.** Let *W* be a set of (possibly ine cient) functions. A *W-indexed* primitive *P*[*W*] is a *set* of primitives $fP[W]g_{W2W}$ indexed by W *2 W* where, for each W *2 W*, *P*[W] = (*F*[W], *R*[W]) is a primitive according to De nition 3.1.1.

For the special case of $\mathcal{W} = \{W\}$ we get back the standard primitive of Definition 3.1.1. We will now define variations of of indexed primitives that restrict the family to a smaller class $\mathcal{W}'$ and, for every $W \in \mathcal{W}'$, it might further restrict the set of correct implementations to be a subset of $F[W]$. We first define restricted forms of indexed primitives then provide various restrictions that will be of interest to us.

Definition 6.5.2 (Restrictions of indexed primitives)**.** For $P[\mathcal{W}] = \{(F[W], R[W])\}_{W \in \mathcal{W}}$ and $P'[\mathcal{W}'] = \{(F'[W], R'[W])\}_{W \in \mathcal{W}'}$, we say $P'[\mathcal{W}']$ is a *restriction* of $P[\mathcal{W}]$ if the following conditions hold: (1) $\mathcal{W}' \subseteq \mathcal{W}$, and (2) for all $W \in \mathcal{W}'$, $F'[W] \subseteq F[W]$, and (3) for all $W \in \mathcal{W}'$, $R'[W] = R[W]$.

Definition 6.5.3 (Efficient restrictions)**.** We call a restriction $P'[\mathcal{W}']$ of $P[\mathcal{W}]$ an *efficient* restriction if $\mathcal{W}' = \{w\}$ where w is is a polynomial time algorithm (with no oracle calls). In this case, we call $P'[w]$ simply a w-restriction of $P[\mathcal{W}]$.

We are particularly interested in indexed primitives when they are indexed by the universal algorithm for circuit evaluation. This is the case for all the primitives of witness encryption, predicate encryption,[2] fully homomorphic encryption, and IO. All of the examples of the special primitives discussed in previous section fall into this category. Finally, the formal notion of what we previously simply called a `special' primitives is defined as follows.

Definition 6.5.4 (The universal variant of indexed primitives)**.** We call $P'[\{w\}]$ the *universal variant* of $P[\mathcal{W}]$ if $P'[\{w\}]$ is an efficient restriction of $P[\mathcal{W}]$ for the specific algorithm $w(\ )$ that interprets its input as a pair (x, C) where C is a circuit, and then it simply outputs $C(x)$.

For example, in the case of witness encryption, the relation between witness w and attribute a is verified by running a as a circuit over w and outputting the first bit of this computation. In order to generalize the notion of universal variants of indexed primitives (i.e., special primitives for short) we need the following definition.

Definition 6.5.5 ($w^{(\cdot)}$-restrictions)**.** For an oracle algorithm $w^{(\cdot)}$ we call the indexed primitive $P'[\mathcal{W}'] = \{(F'[W], R[W])\}_{W \in \mathcal{W}'}$ the $w^{(\cdot)}$-restriction of $P[\mathcal{W}] = \{(F[W], R[W])\}_{W \in \mathcal{W}}$, if $P'[\mathcal{W}']$ is constructed as follows. For all $W \in \mathcal{W}$ and F, we include $W \in \mathcal{W}'$ and $F \in F'[W]$, if it holds that $W = w^F$ and $F \in F[W]$.

Definition 6.5.6 (The monolithically-extended variant of indexed primitives)**.** We call $P'[\mathcal{W}']$ the *monolithically-extended* variant of $P[\mathcal{W}]$ if $P'[\mathcal{W}']$ is a $w^{(\cdot)}$-restriction of $P[\mathcal{W}]$ for the specific $w^{(\cdot)}$ that interprets its input (x, C) as a pair where $C^{(\cdot)}$ is an *oracle-aided* circuit such that $w(x, C)$ executes and outputs $C^{(\cdot)}(x)$ by forwarding all of C's oracle queries to its own oracle.

[2] Even in this case, we can imagine that we are running a circuit on another input and take the first bit of it as the predicate.

Remark 6.5.7. While one may de ne various other forms of extensions for a primitive where $w^{(\cdot)}$ is not necessarily a universal circuit, we will mostly be concerned with the monolithically-extended variants of primitives as de ned above. As such, for the sake of brevity, whenever we refer to the extended variant of primitives we mean the monolithically-extended primitive (unless explicitly noted otherwise).

Remark 6.5.8 (Non-black-box relation between the universal and extended variants of *P*)**.** We emphasize that even though the universal variant $P^{\emptyset}[w]$ and the extended variant $P^{\emptyset}[W^{\emptyset}]$ of *P* are tightly related, they are indeed di erent cryptographic primitives according to the [RTV04] framework, simply because their input formats are di erent. However, one can get one from the other one in a *non-black-box* way. Moreover, when we are in a relativized world where an *ine cient* oracle O is available for free, $P^{\emptyset}[W^{\emptyset}]$ could also have planted O gates in its input circuit, while this is not allowed in the universal variant $P^{\emptyset}[w]$ since w is not an oracle algorithm.

Case of witness encryption. Here we show how to derive the de nition of extended witness encryption as a special case. First note that witness encryption's decryption is indexed by an algorithm $V(w, a)$ that could be any predicate function. In fact, it could be any function where we pick its rst bit and interpret it as a predicate. So WE is indeed indexed by $V \in \mathcal{V}$ which is the set of all predicates. Then, the standard de nition of witness encryption for circuit satis ability (which is the most powerful WE among them all) is simply the universal variant of this indexed primitive WE[*V*], and the following will be exactly the de nition of the (monolithically) extended variant of WE[*V*], which we simply call extended WE.

Definition 6.5.9 (Extended Witness Encryption)**.** Let $\mathsf{V}^{(\mathsf{Enc},\mathsf{Dec})}(w, a)$ be the `universal circuit-evaluator' Turing machine, which is simply an algorithm with oracle access to subroutines (Enc, Dec) that interprets a as a circuit with possible (Enc, Dec) gates and runs a on w and forwards any oracle calls made by a to its own oracle and forwards the answer back to the corresponding gate inside a to continue the execution. The *extended witness encryption* scheme (de ned by V) consists of two PPT algorithms $(\mathsf{Enc}, \mathsf{Dec}_{\mathsf{V}})$ de ned as follows:

> $\mathsf{Enc}(a, m, 1^{\kappa})$: is a randomized algorithm that given an instance $a \in \{0,1\}^*$ and a message $m \in \{0,1\}^*$, and security parameter κ (and randomness as needed) outputs $c \in \{0,1\}^*$.
>
> $\mathsf{Dec}_{\mathsf{V}}(w, c)$: given ciphertext c and \witness" string w, it either outputs a message $m \in \{0,1\}^*$ or $?$.
>
> Correctness and security are de ned similarly to De nition 2.6.1. But the key point is that here the relation $\mathsf{V}^{(\mathsf{Enc},\mathsf{Dec})}$ is somehow recursively depending on the $(\mathsf{Enc}, \mathsf{Dec} = \mathsf{Dec}_{\mathsf{V}})$ on smaller input lengths (and so it is well de ned).

Definition 6.5.10 (Extended variants of other speci c primitives)**.** Extended variants for the following primitives are de ned similarly to De nition 6.5.9 by allowing their special subroutine to be based on an oracle algorithm that is the universal circuit evaluator:

For attribute-based or predicate encryption, we allow the predicate algorithm P(k, a) to be an oracle algorithm that interprets k as an oracle-aided circuit and runs a on k.

For FHE and Spooky encryption we allow the evaluation function F$(f, m_1, ..., m_t)$ to be an oracle algorithm that that interprets f as an oracle-aided circuit and runs f on $(m_1, ..., m_t)$.

6.6 Monolithic Constructions

We are nally ready to de ne our monolithic framework. Here we assume that for a primitive P we have already de ned what its extended variant $\widetilde{P}$ means.

Definition 6.6.1 (Monolithic Constructions { General Case)**.** Suppose Q is a primitive and $\widetilde{P}$ is the extended variant of the primitive P. Any fully black-box construction for Q from $\widetilde{P}$ (i.e. the extended variant of P) is called a *monolithic* construction of Q from P.

Examples. Below are some examples of *non-black-box* constructions in cryptography that fall into the monolithic framework of De nition 6.6.1.

Gentry's bootstrapping construction [Gen09] plants FHE's own decryption in a circuit for the evaluation subroutine. This trick falls into the monolithic framework since planting gates inside evaluation circuits is allowed.

The construction of IO from functional encryption by [AJ15,BV15] uses the encryption oracle of the functional encryption scheme inside the functions for which decryption keys are issued. Again, such *non-black-box* technique does fall into our monolithic framework.

Definition 6.6.2 (Formal De nition of Monolithic Constructions for Speci c Primitives)**.** Let P be any of the following primitives: attribute, predicate, functional, or witness encryption, FHE, multi-key FHE, or spooky encryption. Then a monolithic construction using P is de ned by rst de ning the extended variant $\widetilde{P}$ for primitive P according to De nitions 6.5.9 and 6.5.10, then applying De nition 6.6.1 (which is based on previously de ned notion of *fully* black-box constructions).

The following transitivity lemma (which is a direct corollary to the transitivity of fully black-box constructions) allows us to derive more impossibility results.

Lemma 6.6.3 (Composing monolithic constructions)**.** *Suppose P, Q, R are cryptographic primitives and Q, P are special primitive and $\widetilde{Q}$ is the extended version of Q. If there is a monolithic construction of $\widetilde{Q}$ from P and if there is a monolithic construction of R from Q, then there is a monolithic construction of R from P.*

Proof. Since there is a monolithic construction of R from Q, by Definition 6.6.1 it means that there exists a extended variant $\widetilde{Q}$ of Q such that there is a fully black-box construction of R from $\widetilde{Q}$. On the other hand, again by Definition 6.6.1, for any extended variant of Q, and in particular $\widetilde{Q}$, there is a fully black-box construction of $\widetilde{Q}$ from some extended variant $\widetilde{P}$ of P. Therefore, since fully-black-box constructions are transitive under nested compositions, there is a fully construction of R from $\widetilde{P}$ which (by Definition 6.6.1) means that we have a monolithic construction of R from P. □

Getting more separations. A corollary of Lemma 6.6.3 is that if one proves: (a) There is no monolithic construction of R from P and (b) there *is* a monolithic construction of any extended variant $\widetilde{R}$ (of R) from Q, then these two together imply that: there is no monolithic construction of Q from P. We will use this trick to derive our impossibility results from a core of two separations regarding variants of witness encryption.

Chapter 7

Monolithic Separation of IO from All-or-Nothing Encryption Primitives

7.1 Introduction

After clearly de ning our new monolithic framework, we can nally begin to use it to prove our separations of IO from more expressive primitives. Speci cally, the main result of this chapter proves that several powerful (so-called \all-or-nothing") encryption primitives such as predicate encryption and fully-homomorphic encryption are incapable of producing IO via a *monolithic* construction as described in Chapter 6. A summary of our results is presented in Figure 7.1. More speci cally, we prove the following theorem.

Theorem 7.1.1 (Main Result)**.** *Let P be one of the following primitives: fully-homomorphic encryption, attribute-based encryption, predicate encryption, multi-key fully homomorphic encryption, or spooky encryption. Then, assuming one-way functions exist and* $\mathbf{NP} \not\subseteq \mathbf{coAM}$*, there is no construction of IO from P in the monolithic model where one is* allowed *to plant P gates arbitrarily inside the circuits that are given to P as input.*

All-or-nothing aspect. One common aspect of *all* of the primitives listed in Theorem 7.1.1 is that they have an all-or-nothing nature. Namely, either someone has the right key to decrypt a message, in which case they can retrieve *all* of the message, or if they do not have the right key then they are supposed to learn nothing. In contrast, in a functional encryption scheme (a primitive that does imply IO) one can obtain a key k_f for a function f that allows them to compute $f(x)$ from a ciphertext c containing the plaintext x. So, they could legitimately

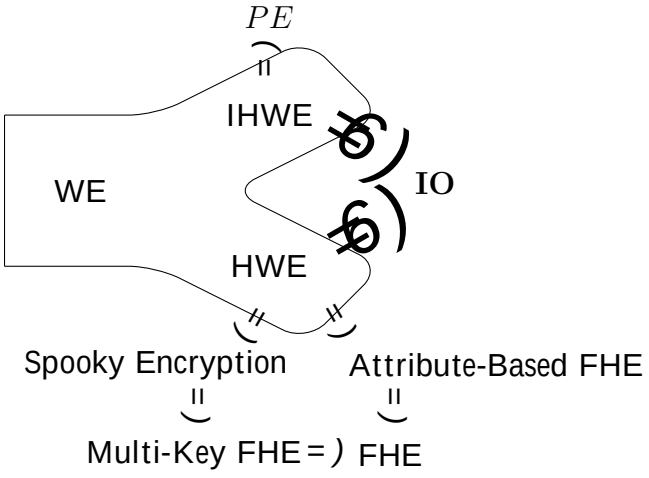

Figure 7.1: Summary of our witness encryption separation results.

learn only a \partial" information about x. Even though we do not yet have a general result that handles such primitives uniformly in one shot, we still expect that other exotic encryption primitives (that may be developed in the future) that are of the all-or-nothing avor will also not be enough for realizing IO. Additionally, we expect that our techniques will be useful in deriving impossibility results in such case.

What does our results say about learning with errors (LWE)? Even though our separations of Theorem 7.1.1 covers most of the powerful LWE-based primitives known to date, it does not imply whether or not we can actually base IO on LWE, which is considered a standard, and hence a desirable, assumption [Reg05]. In fact, our result only rules out speci c paths from LWE toward IO that would go through either of the primitives listed in Theorem 7.1.1. Whether or not a direct construction from LWE to IO is possible still remains as a major open problem in this area.

Key Role of Witness Encryption. Witness encryption and its variations play a key role in the proof or our impossibility results. Speci cally, we consider two (incompatible) variants of WE | namely, instance hiding witness encryption and homomorphic witness encryption. The rst notion boosts the security of WE and hides the statement while the second enhances the functionality of WE with some homomorphic properties. We obtain our separation results in two steps. First, we show that neither of these two primitives imply IO in a monolithic way. Next, we show that these two primitives imply extended versions of all the all-or-nothing primitives listed above in a monolithic way. The nal separations follow from a speci c transitivity lemma that holds in the monolithic model.

As described above, to derive more separation lower bounds for IO, e.g., from a primitive Q, we could de ne a \middle primitive" P and the following: (1) show that P does not imply IO in the monolithic model, and (2) show that there is a monolithic construction of the *extended* version of Q from P. In this work we will use variants of WE to play the role of this middle primitive P. Namely we will use the following two variants: (1) \instance-hiding" WE, and (2) \instance-revealing homomorphic" WE.[1] Below we will discuss these notions in more details, but before doing that we shall point out that these two notions are *incomparable* strengthenings of the basic notion of witness encryption. The rst one strengthens witness encryption in terms of security (which is essential for achieving predicate encryption) while the latter strengthening is in terms of functionality (which is essential for achieving full-homomorphic encryption and its variants). These two notions are incomparable and in fact strengthen the standard notion of witness encryption in somewhat incomparable ways. As a side remark, we note that strengthening witness encryption to incorporate instance hiding property precludes us from also having compact homomorphism along with. Therefore, we need to consider these two primitives separately. Furthermore, showing that obfuscation is

[1] In fact, to be able to use these primitives to derive other primitives we will need these WE variants to have some \extractability" properties as well. However, in this technical overview we will not focus on that aspect and will describe only the main ideas through the simpler (non-extractable) versions of these primitives.

not possible from either of them poses different technical challenges.

In the following we will first focus on the simpler and basic case of separating IO from WE. This will allow us to communicate some basic tools that we use from previous work and also discuss some of our new ideas, and then we will turn into the specific variants of WE that we mentioned and explain the new ideas that each case requires. However, we first quickly recap the recent developments in previous work that provides us with a recipe of how to prove *fully* black-box lower bounds for IO. As we discussed above, that framework could *still* be used by us to prove lower bounds for IO in the monolithic model, but we have to do so with respect to the stronger variants of our primitives of interest (e.g., WE).

7.1.1 Known Recipe for Proving Lower-bounds for IO

Using the techniques and results of Section 5.4 we can arrive at the following (informally stated) lemma which provides us with an abstract method for proving fully black-box (and monolithic) lower bounds for IO.

Lemma 7.1.2 (Proving Lower Bounds for IO (Informal))**.** *Suppose I is an idealized oracle and suppose we can "compile out" I from any IO construction in a world where I is accessible to get an IO scheme in the plain model where the new scheme is only approximately correct: it computes the correct answer only over* $99/100$ *fraction of the input. Also suppose primitive P can be securely implemented in the randomized model I. Then there is no fully black-box construction of IO from P.*

See Section 7.2.1 (particularly Lemma 7.2.7 there) for details on how to derive this lemma in conjunction with previous works. Below we discuss how we apply this approach to the case of variants of witness encryption.

7.1.2 Warm-Up: The Basic Case of Witness Encryption

To separate IO from the *extended* variant of WE (i.e., separating IO from WE in the monolithic model) we can try to apply Lemma 7.1.2 and prove a *fully* black-box separation for IO from the *extended* variant of WE. Suppose V is the oracle universal circuit algorithm that defines the witness verification of the *extended* WE primitive we would like to separate from IO. As a first try, let see what happens if we try to use the following simple idealized oracle to implement this extended WE primitive.

The Idealized WE Oracle I. We use a random injective oracle $\mathsf{Enc}(a, m) \rightarrow c$ to encrypt any message m under the attribute/instance a. The other oracle subroutine $\mathsf{Dec}_{\mathsf{V}}(w, c)$ does the decryption with respect to V as follows:

1. If $\nexists\, x$ such that $\mathsf{Enc}(x) = c$, output $\perp$. Otherwise:
2. Find x such that $\mathsf{Enc}(x) = c$ and parse it as $x = (a, m)$.

3. If V (w, a) = 0 output *?*. (Note that this step might need launching some recursive oracle calls.) Otherwise, output m.

Now our goal is two-fold: showing that this oracle indeed gives us a secure implementation of the *extended* WE primitive (de ned by relation V) and that the oracle *I* can be securely compiled out of any IO construction (while keeping the approximate correctness). Proving that this oracle gives us a secure extended WE, while requiring an analysis that is quite involved, is often safe to believe is true due to the ideal nature of *I*. Thus, the challenge lies in the second goal of proving that we can compile out *I* from an IO scheme while preserving correctness, and so we will discuss this in more detail.

Trying to Compile out *I* from any IO scheme IO^I. Let IO^I be an obfuscation in some idealized model *I* that implements the extended WE primitive. As mentioned above, one of our main tasks is to \compile-out" the oracle from an ideal model secure obfuscator to get an approximate plain-model obfuscator IO^0 that is also secure. To do so we use the ideas introduced in [CKP15] and those developed in Chapter 4 which is based on learning the \heavy queries" asked by the obfuscated code to the oracle and hard-coding them into the obfuscated code in the plain model.[2] Speci cally, the new plain-model obfuscator IO^0, given a circuit C to obfuscate would work in two steps. The rst step of IO^0 is to emulate $\mathsf{IO}^I(C)$ to get an ideal-model obfuscation B, making sure to lazily evaluate (i.e., emulate) any queries issued to *I*. The second step of IO^0 is to learn the queries that are \likely" to be asked by $B^I(x)$ for a random input x. We do this by executing $B^I(x_i)$ (while we emulate the queries to *I* consistently) enough number of times for di erent x_i in an e ort to learn all the highly probable queries, which we denote by Q_B. The output of IO^0 is the plain-model obfuscation $B^0 = (B, Q_B)$, where B is the ideal-model obfuscation and Q_B is the set of learned queries. To evaluate the obfuscation over a new random input x, we simply execute $B^0(x) = B^I(x)$ while emulating any queries to *I* consistently relative to Q_B.

Security of compiler: only include simulatable information in B^0. The security of the new plain-model obfuscator crucially depends on whether the queries Q_B that we publish are simulatable. In other words, we want to prove that if the adversary A against this new plain-model does not gain any additional advantage because these queries can be simulated by having A itself run $B^I(x_i)$ several times. As a result, we will *only* put queries in Q_B (forwarded to be part of the plain-model obfuscator) where these queries could be obtained by the adversary as well. Doing so allows us to immediately reduce the security to that of the ideal-model obfuscation. It is therefore important to release only those queries that can be simulated to ensure security is preserved (for example, we cannot publish the queries asked during the emulation in step 1 as they are not simulatable).

The other task we will have is to prove the *approximate* correctness of the new model, which informally states that an execution $B^0(x)$ should be correct with high probability

[2] Note that this compiling out process is not independent of the oracle being removed since di erent oracles may require di erent approaches to be emulated. However, the general high-level idea is the same.

over x. Note that unless we ask an unlearned query to I that was previously asked by some hidden part of the emulation process (e.g. during step 1), the execution should be statistically distributed to an ideal execution of $B^I(x)$ where the queries are not emulated but answered using a real oracle. In fact if the oracle I did not involve any \internal/hidden" query asked by V, we could use the arguments given in Chapter 4 to show that the probability that we ask an unlearned hidden query occurs with su ciently small probability.

Main challenge for proving the correctness A subtle, but extremely important point here that prevents us from proving the approximate correct is that when we run a decryption query of the form $\mathsf{Dec}_V(w, c)$ in the actual ideal model, we will *not* see what queries the veri er V asks from the oracle itself. That is the reason that we call any such query a \hidden" or \indirect" query (in eyes of the person asking the query $\mathsf{Dec}_V(w, c)$ in the ideal model). Therefore, for sake of security we are *not* allowed to provide this information to the nal plain-model obfuscated code, even though we have emulated all of these oracle query/answers from the beginning on our own! This is exactly the reason that the approximate correctness of the nal plain-model obfuscated code B' could be risked, because we might no longer be able to continue emulating the oracle consistently while executing $B'(x)$ on a random point x.

Resolving the challenge. The way we handle the above issue of nding the heavy queries, even if they are of the hidden/indirect type is as follows. We will make them to be clear! We will do so through a new subroutine in the oracle that will always reveal the instance/attribute a inside a plaintext c. Namely, we add the following subroutine to I:

$\mathsf{Rev}(c)$: given ciphertext c outputs a for which $\mathsf{Enc}(a, m) = c$.

Now, we can pretend that any algorithm who wants to call a decryption query of the form $\mathsf{Dec}_V(w, c)$ to our oracle, it will rst obtain the relevant instance/attribute $a = \mathsf{Rev}(c)$, run the veri er $V(a, m)$ on its own, and if the test passes $V(a, m) = 1$ the algorithm will indeed ask the query $\mathsf{Dec}_V(w, c)$. This is a simple \canonicalization" of the algorithms in the idealized model I, however this will be enough to guarantee that no internal/indirect query asked by V during the computation of $\mathsf{Dec}_V(w, c)$ remains hidden! In other words, by running the obfuscated code on enough number of random points, we will be able to discover all the heavy queries that are needed for a successful execution of the new obfuscated code in the plain model.

We shall nally emphasize that the above trick of adding a new subroutine $\mathsf{Rev}(c)$ does *not* violate the security of (extended) WE relative to our oracle I.

7.1.3 Separating IO from Instance-Hiding WE

To derive our separations for primitives such as predicate encryption, we will rst separate IO from a variant of WE which we call \instance hiding" WE, and we also show that this

primitive is strong enough to imply (even extended) PE in a monolithic way.[3] An instance hiding WE is a variant of WE where two ciphertexts Enc(a_0, m_0) = c_0, Enc(a_1, m_1) = c_1 that hide *di erent* instance/attributes a_0 6= a_1 are indistinguishable so long as there is no witness that satis es either of these instances; namely V(a_b, w) = 0 for all w and b 2 f0, 1g (see De nition 2.6.3). The reason that we will need such variant of WE for constructing PE is that PE itself has the same nature and di erent ciphertexts under unsatis able attributes should remain indistinguishable.

Challenge: we cannot have Rev( ) as part of oracle. It becomes immediately clear that we cannot use the same oracle of the previous section (for the case of basic WE) to prove our separation for IHWE anymore. The reason is that the subroutine Rev(c) = a clearly destroys the instance hiding property that we want to keep!

Idea: revealing attribute a conditionally, at decryption time. The new idea that we will introduce in for the case of IHWE is that we will still \weaken" the security (towards helping the discovery of the hidden queries asked by V(w, a) for a decryption oracle query). However, we will do so in a conditional way so that it will not contradict the instance hiding property. Namely, we will only reveal a if the decryption succeeds. More formally, we will use the following decryption subroutine. Dec$_V$(w, c) does the decryption with respect to V as follows:

1. If @ x such that Enc(x) = c, output *?*. Otherwise:
2. Find x such that Enc(x) = c and parse it as x = (a, m).
3. If V (w, a) = 0 output *?*. (Note that this step might launch some recursive oracle calls.) Otherwise, output *both* of (a, m).

Note that the above modi ed way of decryption ciphertexts will *not* interfere with the instance hiding property of the scheme. However, it will be hugely important for us to still be able to discover the heavy queries of the obfuscated code in the idealized model (and so we can still compile out the idealized oracle from any IO construction). Unfortunately, we are not able to discover all the hidden/internal queries of a decryption query immediately, but we can do so only conditionally. Speci cally, whenever the obfuscated circuit calls a decryption query Dec(w, c) and the underlying message x is successfully decrypted, the obfuscated circuit can now run V (w, a) = 1 to reveal the queries asked. This can be thought of as a weaker form of canonicalization of the code of the veri er that still *sometimes* reveals the hidden parts of the oracle computation. That is because, it is possible that the decryption fails, in which case we may have some hidden queries Q_V asked by an underlying V (w, a) = 0 that cannot be revealed or learned. Nonetheless, we show using a careful argument that by doing enough rounds of learning (over random inputs) and \opening up" the internal hidden

[3] To be more formal, we need this primitive to be \extractable" instance hiding WE, but for sake of simplicity we defer the extractability issue to the later sections where we de ne everything formally.

parts of the veri cation queries as much as possible, we will still get enough information to run the nal (plain model) obfuscated code B' with approximate correctness close to 1. We refer the reader to the next sections for formal proofs.

7.1.4 Separating IO from Homomorphic WE

For obtaining our separations for IO from the set of \homomorphic" primitives listed in Theorem 7.1.1 we will employ a \middle" primitive called \homomorphic (instance revealing) witness encryption". Roughly speaking, this primitive allows homomorphic operations over the ciphertexts that are encrypted under the *same* attribute/instance. However, it is possible to always extract this attribute e ciently (this is exactly how we modi ed the witness encryption in our basic case to be able to prove the separation from IO). In other words, we will need an extra functionality over the instance revealing WE as described in previous section.

New challenge: homomorphic queries. Due to the fact that we are still working with an *instance revealing* form of WE, this simpli es our job and we will *not* fall into the challenges that we faced for the case of Instance-hiding WE. However, we will have a new challenge: we need to nd the heavy queries that are asked during the *evaluation* procedure! This is something that did not exist in the basic and instance hiding versions of WE. In Section 7.5 we will show that, essentially, the same learning procedure will allow us to learn enough information to simulate the last nal (real) execution of the obfuscated code on a given input. In particular, for an evaluation query $\mathsf{Eval}_{\mathsf{F}}(c_1, \ldots, c_k)$ that computes some homomorphic evaluation $\mathsf{F}(m_1, \ldots, m_k)$ over the plaintexts inside $(c_1, \ldots, c_k)$, there are two cases: either we already have learned the messages $m_1, \ldots, m_k$ in which case we can also run the algorithm $\mathsf{F}(m_1, \ldots, m_k)$, but we do not know *any* of these messages, we will simply generate a random answer and use it to emulate the answer to the homomorphic operation. A careful analysis is needed to prove that this in fact leads to an approximately correct execution of the code in the plain model.

7.1.5 Primitives Implied by Our Variants of WE

We now describe our ideas on how the previously described variants of WE can be used to imply extended PE and extended FHE. Building on similar principles we can also obtain extended spooky encryption and extended attribute-based FHE. In realizing these primitives we use the extended and extractable versions of these variants of WE. The constructions are based on ideas developed to demonstrate how WE [GGSW13] and extractable WE [BCP14, ABG+13] can be used to construct various exotic cryptographic primitives.

Getting Extended PE from Instance Hiding WE

We start by showing how instance-hiding WE implies extended selectively-secure PE for the predicate P(k, a) where k *2* K and a *2* A.[4] The idea is straightforward. A secret key for a string k is just a signature on it and a ciphertext encrypting a message x = (a, m) is an (instance hiding) witness encryption with respect to the attribute C that represents a Boolean circuit for verifying the signature and checking that $P(k, a)$ = 1.

To decrypt the ciphertext, we use the decryption of the (instance hiding) witness encryption that takes as input the signature sk_k (acting as a key for the string k) and outputs m if $C(k, sk_k, a)$ = 1. Security of this construction follows directly from the security of the extractable instance-hiding witness encryption scheme and the existential unforgeability of the underlying signature scheme. More speci cally, by security of witness encryption we have that any adversary distinguishing witness encryption ciphertexts can be used to recover a witness which our case serves as an existential forger. Note that P is allowed to have gates of the PE subroutines planted in them. In the main body, we argue that the above described construction supports this.

Getting Extended FHE from Homomorphic WE

Realizing extended FHE from instance-revealing homomorphic witness encryption is essential the same as the Garg et al. [GGSW13] construction of public-key encryption from WE and PRG. The public-key for the FHE is just the output PK = $G(s)$ of a PRG G on input a seed s and ciphertext is just a witness encryption ciphertext encrypting m for V where V(w, C_{PK}) = 1 if and only if $C_{PK}(w)$ = 1 where $C_{PK}(w)$ checks if PK = $G(w)$, outputs 1 if it is the case and 0 otherwise. Now by the homomorphic property of the witness encryption this new encryption scheme is also homomorphic. On the other hand, security follows directly from the security of the PRG and the WE scheme. Note that the evaluation procedure of the homomorphic encryption scheme is allowed to have gates of its subroutines planted in it. In the main body, we argue that the above described construction supports this.

7.2 Approach for Proving Lower Bounds on IO

In this section we provide a general approach for proving lower bounds on IO schemes under a given idealized model. We develop this approach based on the techniques used in Chapter 4 where we proved the impossibility of VBB obfuscation under di erent idealized models and Section 5.4 where we showed how to prove fully black-box separations of IO using [BBF16]. The abstract blueprint outlined here will later be applied and instantiated in di erent ways during the subsequent chapters when we prove IO separation results from various primitives. Each application of this approach o ers unique challenges that depends on the underlying primitive that we would want to separate IO from, and we will address these challenges in their respective upcoming chapters.

[4] Details on strengthening the result to full-security have been postponed to the main body.

7.2.1 General Approach

Our techniques that we will develop for proving separations for IO rely on the process of \compiling out" an idealized oracle I from an IO construction. Suppose that we can indeed remove the oracle from the obfuscator in a manner similar to how we did in Chapter 4. Since we know that *statistically* secure IO does not exist in the *plain* model [GR07] this indicates that perhaps we can compose the two steps and get a query-e cient attacker against IO in the idealized model I. The more accurate line of argument is more subtle and needs to work with *approximately correct* IO and uses a recent result of Brakerski, Brzuska, and Fleischhacker [BBF16] who ruled out the existence of statistically secure approximate IO.

To formalize the notion of \compiling out" an oracle in more than one step we need to formalize the intuitive notion of sub oracles in the idealized/randomized context.

Definition 7.2.1 (Sub-models)**.** We call the idealized model/oracle O a sub-model of the idealized oracle I with subroutines $(I_1, \ldots, I_k)$, denoted by $O \vee I$, if there is a (possibly empty) $S \subseteq \{1, \ldots, k\}$ such that the idealized oracle O is sampled as follows:

> First sample $I \leftarrow I$ where the subroutines are $I = (I_1, \ldots, I_k)$.
>
> Then provide access to subroutine I_i if and only if $i \in S$ (and hide the rest of the subroutines from being called).

If $S = \emptyset$ then the oracle O will be empty and we will be back to the plain model.

Definition 7.2.2 (Simulatable Compiling Out Procedures for IO)**.** Suppose $O \not\vee I$. We say that there is a simulatable compiler from IO in idealized model I into idealized model O with correctness error ϵ if the following holds. For every implementation $P_I = (\mathsf{iO}_P, \mathsf{Ev}_P)$ of δ-approximate IO in idealized model I there is a implementation $P_O = (\mathsf{iO}_O, \mathsf{Ev}_O)$ of $(\delta + \epsilon)$-approximate IO in idealized model O such that the only security requirement for these two implementations is that they are related as follows:

> Simulation: There is an e cient PPT simulator S and a negligible function $\mu(\cdot)$ such that for any C:
>
> $$(S(\mathsf{iO}^I(1^\kappa, C)), \mathsf{iO}^O(1^\kappa, C)) \leq \mu(\kappa)$$
>
> where $(\cdot, \cdot)$ denotes the statistical distance between random variables.

The above de nition generalizes the VBB simulatability requirement that a compiler possessed in Theorem 4.4.5 to the setting of IO. It is easy to see that the existence of the simulator according to De nition 7.2.2 implies that P_O in idealized model O is \as secure as" P_I in the idealized model I since a compiler that guarantees such a simulation property would also imply indistinguishability security for the case of IO. Namely, any oracle-mixed attacker against the implementation P_O in model O with advantage δ (over an in nite sequence of security parameters) could be turned in to an attacker against P_I in model I that breaks against P_I with advantage $\delta - \mathsf{negl}(\kappa)$ over an in nite sequence of security

parameters. Therefore one can compose the compiling out procedures for a constant number of steps (but not more, because there is a polynomial blow up in the parameters in each step).

By composing a constant number of compilers and relying on the recent result of Brakerski, Brzuska, and Fleischhacker [BBF16] one can get a general method of breaking IO in idealized models. We first state the result of [BBF16].

Theorem 7.2.3 (Impossibility of Approximate Statistically-secure IO [BBF16])**.** *Suppose one-way functions exist,* $\mathbf{NP} \not\subseteq \mathbf{coAM}$*, and* δ, ϵ: $\mathbb{N} \to [0,1]$ *are such that* $2\epsilon(\kappa) + 3\delta(\kappa) < 1 - 1/\mathrm{poly}(\kappa)$*. Then there is no* (ϵ, δ)*-approximate statistically-secure CIO (see Definition 2.5.4) for all poly-size circuits.*

The above theorem implies that if we get any implementation for IO in the plain model that is $1/100$-approximately correct, then there is a computationally unbounded adversary that breaks the *statistical* security of IO with advantage at least $1/100$ over an infinite sequence of security parameters. Using this result, the following lemma shows a way to obtain attacks against IO in idealized models.

Lemma 7.2.4 (Attacking IO Using Nested Oracle Compilers)**.** *Suppose* $\Pi = \Pi_0 \vee \Pi_1 \vee \cdots \vee \Pi_k = \Pi$ *for constant* $k = O(1)$ *are a sequence of idealized models. Suppose for every* $i \in [k]$ *there is a simulatable compiler for IO in model* Π_i *into model* Π_{i-1} *with correctness error* $\epsilon_i < 1/(100k)$*. Then, assuming one-way functions exist,* $\mathbf{NP} \not\subseteq \mathbf{coAM}$*, any implementation* P *of IO in the idealized model* Π *could be oracle-mixed broken by a polynomial-query adversary* A *with a* constant *advantage* $\delta > 1/100$ *for an infinite sequence of security parameters.*

Proof. Starting with our initial ideal-model construction $P_\Pi = P_{\Pi_k}$, we iteratively apply the simulatable compiler to get $P_{\Pi_{i-1}}$ from P_{Π_i} for $i = \{k, \ldots, 1\}$. Note that the final correctness error that we get is $\epsilon_{\Pi_0} < k/(100k) < 1/100$, and thus by Theorem 5.4.3 there exists a computationally unbounded attacker A_{Π_0} against P_{Π_0} with constant advantage δ. Now, let S_i be the PPT simulator whose existence is guaranteed by Definition 7.2.2 for the compiler that transforms P_{Π_i} into $P_{\Pi_{i-1}}$. We inductively construct an adversary A_{Π_i} against P_{Π_i} from an adversary $A_{\Pi_{i-1}}$ for $P_{\Pi_{i-1}}$ starting with A_{Π_0}. The construction of A_{Π_i} simply takes its input obfuscation in the Π_i ideal-model iO^{Π_i}, runs $S_i(\mathrm{iO}^{\Pi_i})$ and feeds the result to $A_{\Pi_{i-1}}$ to get its output. Note that, after constant number k, we still get $\delta' < \delta - k \cdot \mathrm{negl}(\kappa)$ a constant advantage over infinite sequence of security parameters against P_{Π_k}. □

Using a variant of the Borel–Cantelli lemma, we can in fact prove that oracle-mixed attacks with *constant* advantage lead to breaking oracle-fixed constructions.

Lemma 7.2.5. *If there is an algorithm* A *that oracle-mixed breaks a construction* P^Π *of* P *in idealized model* Π *with advantage* $\epsilon(\kappa) > 1/\mathrm{poly}(\kappa)$ *for an infinite sequence of security parameters, then the same attacker* A *oracle-fixed breaks the same construction* P^Π *over a (perhaps more sparse but still) infinite sequence of security parameters.*

Proof. Let A be an oracle mixed attacker against P in the idealized model I with advantage $\epsilon(\kappa)$ and de ne $E^{P}_{A,}$ to be the event that A succeeds in breaking security of P over security parameter κ. Then we have that for in nitely many security parameters κ:

$$\Pr_{A;I}[E^{P}_{A,}] \quad \epsilon(\kappa)$$

Note that since this is an oracle-mixed attacker, the probability is over the randomness of the adversary, the experiment *and the oracle* I. Now de ne functions ϵ' and ϵ'' such that $\epsilon'' = \epsilon \quad \epsilon'$ (1). Then, by an averaging argument we get the following:

$$\Pr_{I}[\Pr_{A}[E^{P}_{A,}] \quad \epsilon'(\kappa)] \quad \epsilon''(\kappa)$$

Let D_i be the event that $\Pr_A[E^{P}_{A;i}] \quad \epsilon'$. From the above we can see that $\Pr_I[D_i] \quad \epsilon''(\kappa)$ for every i 2 N. Now, since $\epsilon''(\kappa)$ (1), by Lemma 2.2.2 we can conclude that with probability $\epsilon''(\kappa)$ over the choice of I, D_i happens for in nitely many i. Thus, since there exists non-zero (in fact, constant) measure of oracles that A can break successfully over in nitely many security parameters, we can x any oracle out of those that A can break thus yielding an oracle- xed attack on P. □

The reader may notice that the above lemma has already been implicitly used (and independently proven) in Section 5.4 for the case that P is the IO primitive. Indeed Lemma 7.2.5 generalizes this to any primitive and states that such a reduction is independent of the primitive and relies only on the advantage of the attacker.

The next lemma follows as a direct corollary to Lemmas 3.1.8 and 7.2.5, and implies that it su ces to provide a \good" oracle-mixed attacker against Q in ideal model I in order to show a separation of Q from any P that exists relative to I.

Lemma 7.2.6 (Separation Using Idealized Models)**.** *Suppose I is an idealized model, and the following conditions are satis ed:*

> **Proving oracle-fixed security of P.** *There is an oracle xed black-box construction of P relative to I.*
>
> **Breaking oracle-mixed security of Q with (1) advantage.** *For any construction Q^P of Q relative to I there is a computationally-unbounded query-e cient attacker A (whose query complexity is bounded by the level of security demanded by P) such that for an in nite sequence of security parameters $\kappa_1 < \kappa_2 < \ldots$ the advantage of A in oracle-mixed breaking Q^I is at least $\epsilon(\kappa_i)$ (1).*

Then there is no fully black-box construction for Q from P.

Finally, by putting Lemma 7.2.4 and 7.2.6 together we get a lemma for proving black-box lower bounds for IO.

Lemma 7.2.7 (Lower Bounds for IO using Oracle Compilers). *Suppose ? = Π_0 v Π_1 v Π_k = Π for constant $k = O(1)$ are a sequence of idealized models. Suppose for every i 2 $[k]$ there is a simulatable compiler for IO in model Π_i into model $\Pi_{i\ 1}$ with correctness error $\epsilon_i < 1/(100k)$. If primitive P can be oracle- xed constructed in the idealized model Π, then there is no fully black-box construction of IO from P.*

We will indeed use Lemma 7.2.7 to derive lower bounds for IO even in the *monolithic* model by relating such constructions to fully black-box constructions.

7.3 Separating IO from Instance Revealing Witness Encryption

In this section, we formally prove our rst main separation theorem which states that there is no monolithic construction of IO from WE (under believable assumptions). It equivalently means that there will be no fully black-box construction of indistinguishability obfuscation from extended witness encryption scheme.

Theorem 7.3.1. *Assume the existence of one-way functions and that* $\mathbf{NP}$ *6* $\mathbf{coAM}$*. Then there exists no monolithic construction of indistinguishability obfuscation (IO) from witness encryption (WE).*

In fact, we prove a stronger result by showing a separation of IO from a stronger (extended) version of witness encryption, which we call *extractable instance-revealing* witness encryption. Looking ahead, we require the extractability property to construct (extended) attribute-based encryption (ABE) from this form of witness encryption. By using Lemma 6.6.3, this would also imply a separation of IO from extended ABE.

Definition 7.3.2 (Extended Extractable Instance-Revealing Witness Encryption (ex-EIRWE)). Let V be a universal circuit-evaluator Turing machine as de ned in De nition 6.5.9. For any given security parameter κ, an extended *extractable instance-revealing witness encryption* scheme consists of three PPT algorithms P = (Enc, Rev, Dec) de ned as follows:

> Enc$(a, m, 1\ )$: given an instance a *2* $f0,1g$ and a message m *2* $f0,1g$, and security parameter κ (and randomness as needed) it outputs c *2* $f0,1g$.
>
> Rev(c) : given ciphertext c outputs a *2* $f0,1g$ *[* $f?g$.
>
> Dec(w, c) : given ciphertext c and \witness" string w, it outputs a message $m^{\emptyset}$ *2* $f0,1g$.

An extended extractable instance-revealing witness encryption scheme satis es the following completeness and security properties:

> **Decryption Correctness:** For any security parameter κ, any (w, a) such that $V^P(w, a)$ = 1, and any m it holds that
>
> $$\Pr_{\text{Enc;Dec}}[\text{Dec}(w, \text{Enc}(a, m, 1\ )) = m] = 1$$

Instance-Revealing Correctness: For any security parameter κ and any (a, m) it holds that:

$$\Pr_{\mathsf{Enc};\mathsf{Rev}}[\mathsf{Rev}(\mathsf{Enc}(a, m, 1^\kappa)) = a] = 1$$

Furthermore, for any c for which there is no a, m, κ such that $\mathsf{Enc}(a, m, 1^\kappa) = c$ it holds that $\mathsf{Rev}(c) = ?$.

Extractability: For any PPT adversary A and polynomial $p_1(.)$, there exists a PPT (black-box) straight-line extractor E and a polynomial function $p_2(.)$ such that the following holds. For any security parameter κ, for all $a \in \{0,1\}^*$, and any $m_0 \neq m_1$ of the same length $|m_0| = |m_1|$, if:

$$\Pr\left[A(1^\kappa, c) = b \mid b \xleftarrow{\$} \{0,1\}, c \leftarrow \mathsf{Enc}(a, m_b, 1^\kappa)\right] \geq \frac{1}{2} + \frac{1}{p_1(\kappa)}$$

Then:

$$\Pr[E^A(a) = w \wedge V^P(w, a) = 1] \geq \frac{1}{p_2(\kappa)}$$

Given the above definition of ex-EIRWE, we prove the following theorem, which states that there is no fully black-box construction IO from extended EIRWE.

Theorem 7.3.3. *Assume the existence of one-way functions and that* $\mathbf{NP} \not\subseteq \mathbf{coAM}$. *Then there exists no* monolithic *construction of indistinguishability obfuscation from extractable instance-revealing witness encryption.*

Since extended EIRWE implies witness encryption as defined in Definition 2.6.1, Theorem 7.3.1 trivially follows from Theorem 7.3.3, and thus for the remainder of this section we will focus on proving Theorem 7.3.3.

7.3.1 Overview of Proof Techniques

To prove Theorem 7.3.3, we will apply Lemma 7.2.7 for the idealized extended IRWE model (formally defined in Section 7.3.2) to prove that there is no black-box construction of IO from any primitive P that can be oracle-fixed constructed (see Definition 3.1.4) from . In particular, we will do so for P that is the extended EIRWE primitive. Our task is thus twofold: (1) to prove that P can be oracle-fixed constructed from and (2) to show a simulatable compilation procedure that compiles out from any IO construction. The first task is proven in Section 7.3.3 and the second task is proven in Section 7.3.4. By Lemma 7.2.7, this would imply the separation result of IO from P and prove Theorem 7.3.3.

Our oracle, which is more formally defined in Section 7.3.2, resembles an idealized version of a witness encryption scheme, which makes the construction of extended EIRWE straightforward. As a result, the main challenge lies in showing a simulatable compilation procedure for IO that satisfies Definition 7.2.2 in this idealized model.

7.3.2 The Ideal Model

In this section, we de ne the distribution of our ideal randomized oracle.

Definition 7.3.4 (Random Instance-revealing Witness Encryption Oracle)**.** Let V be a universal circuit-evaluator Turing machine (as de ned in De nition 2.1.1) that takes as input (w, x) where $x = (a, m) \in \{0,1\}^n$ and outputs $b \in \{0,1\}$. We de ne the following *random instance-revealing witness encryption* (rIRWE) oracle $\mathcal{O} = (\mathsf{Enc}, \mathsf{Rev}, \mathsf{Dec}_V)$ as follows. We specify the sub-oracle $\mathcal{O}_n$ whose inputs are parameterized by n, and the actual oracle will be $\mathcal{O} = \{\mathcal{O}_n\}_{n \in \mathbb{N}}$.

Enc: $\{0,1\}^n \to \{0,1\}^{2n}$ is a random injective function.

Rev: $\{0,1\}^{2n} \to \{0,1\} \cup \{?\}$ is a function that, given an input $c \in \{0,1\}^{2n}$, would output the corresponding attribute a for which $\mathsf{Enc}(a, m) = c$. If there is no such attribute then it outputs $?$ instead.

Dec_V: $\{0,1\}^s \to \{0,1\}^n \cup \{\perp\}$: Given $(w, c) \in \{0,1\}^s$, $\mathsf{Dec}(w, c)$ allows us to decrypt the ciphertext c and get $x = (a, m)$ as long as the predicate test is satis ed on (w, a). More formally, do as follow:

1. If $\nexists\, x$ such that $\mathsf{Enc}(x) = c$, output $?$. Otherwise, continue to the next step.
2. Find x such that $\mathsf{Enc}(x) = c$.
3. If $V(w, a) = 0$ output $?$. Otherwise, output $x = (a, m)$.

We de ne a query-answer pair resulting from query q to subroutine $T \in \{\mathsf{Enc}, \mathsf{Dec}, \mathsf{Rev}\}$ with some answer β as $(q \mathbin{?}\!\!= \beta)_T$. The oracle $\mathcal{O}$ provides the subroutines for all inputs lengths but, for simplicity, and when n is clear from the context, we use $\mathcal{O} = (\mathsf{Enc}, \mathsf{Rev}, \mathsf{Dec}_V)$ to refer to $\mathcal{O}_n$ for a xed n.

Remark 7.3.5. We note that since V is a universal circuit-evaluator, the number of queries that it will ask (when we recursively unwrap all internal queries to Dec) is at most a polynomial. This is due to the fact that the sizes of the queries that V asks will be strictly less than the size of the inputs to V. In that respect, we say that V has the property of being *extended poly-query*.

7.3.3 Witness Encryption exists relative to

In this section, we show how to construct a semantically-secure extended extractable IRWE for universal circuit-evaluator V relative to $\mathcal{O} = (\mathsf{Enc}, \mathsf{Rev}, \mathsf{Dec}_V)$. More formally, we will prove the following lemma.

Lemma 7.3.6. *There exists a correct and subexponentially-secure oracle- xed implementation (De nition 3.1.4) of extended extractable instance-revealing witness encryption in the ideal oracle model.*

We will in fact show how to construct a primitive (in the oracle model) that is simpler to prove the existence of and for which we argue that it is su cient to get the desired primitive of EIRWE. We give the de nition of that primitive followed by a construction.

Definition 7.3.7 (Extended Extractable One-way Witness Encryption (ex-EOWE))**.** Let V be a universal circuit-evaluator Turing machine (as de ned in De nition 6.5.9) that takes an instance a and witness w and outputs a bit $b \in \{0,1\}$. For any given security parameter κ, an *extended extractable one-way witness encryption* scheme for V consists of the following PPT algorithms P = (Enc, Rev, Dec_V) de ned as follows:

Enc($a, m, 1^\kappa$) : given an instance $a \in \{0,1\}^*$, message $m \in \{0,1\}^*$, and security parameter κ (and randomness as needed) it outputs $c \in \{0,1\}^*$.

Rev(c) : given ciphertext c returns the underlying attribute $a \in \{0,1\}^*$.

$\text{Dec}_V(w, c)$: given ciphertext c and \witness" string w, it outputs a message $m' \in \{0,1\}^*$.

An extended extractable one-way witness encryption scheme satis es the same correctness properties as De nition 7.3.2 but the extractability property is replaced with the following:

Extractable One-Wayness: For any PPT adversary A and polynomial $p_1(.)$, there exists a PPT (black-box) straight-line extractor E and a polynomial function $p_2(.)$ such that the following holds. For any security parameter κ, k = poly(κ), and for all a, if:

$$\Pr\left[A(1^\kappa, c) = m \mid m \xleftarrow{\$} \{0,1\}^k, c \leftarrow \text{Enc}(a, m, 1^\kappa)\right] \geq \frac{1}{p_1(\kappa)}$$

Then:

$$\Pr[E^A(a) = w \wedge V^P(w, a) = 1] \geq \frac{1}{p_2(\kappa)}$$

Construction 7.3.8 (Extended Extractable One-way Witness Encryption)**.** For any security parameter κ and oracle sampled according to De nition 7.3.4, we will implement an extended EOWE scheme P for the universal circuit-evaluator V using = (Enc, Dec_V) as follows:

WEnc($a, m, 1^\kappa$) : Given security parameter 1^κ, $a \in \{0,1\}^*$, and message $m \in \{0,1\}^{n=2}$ where $n = 2\max(|a|, \kappa)$, output Enc(x) where $x = (a, m)$.

WDec(w, c) : Given witness w and ciphertext c, let $x' = \text{Dec}_V(w, c)$. If $x' \neq ?$, parse as $x' = (a', m')$ and output m'. Otherwise, output $?$.

Remark 7.3.9 (From one-wayness to indistinguishability.)**.** We note that the primitive ex-EOWE, which has one-way security, can be used to build an ex-EIRWE, which has security that is indistinguishability-based, through a simple application of the Goldreich-Levin thoerem [GL89]. Namely, to encrypt a one-bit message b under some attribute

a, we would output the ciphertext c = (Enc(a, r_1), r_2, $\langle r_1, r_2 \rangle$ b) where r_1, r_2 are randomly sampled and $\langle r_1, r_2 \rangle$ is the hardcore bit. To decrypt a ciphertext c = (y_1, r_2, y_3) we would run r_1 = Dec(w, y_1), nd the hardcore bit p = $\langle r_1, r_2 \rangle$ then output b = p y_3. We obtain the desired indistinguishability security since, by the hardcore-bit security of the one-way function $\text{Enc}'_a(r_1, r_2)$:= (Enc(a, r_1), r_2), we have (Enc(a, r_1), r_2, $\langle r_1, r_2 \rangle$ 0) (Enc(a, r_1), r_2, $\langle r_1, r_2 \rangle$ 1) for any xed a.

Lemma 7.3.10. *Construction 7.3.8 is a correct and subexponentially-secure oracle- xed implementation (De nition 3.1.4) of extended extractable one-way witness encryption in the ideal oracle model.*

Proof. To prove the security of this construction, we will show that if there exists an adversary A against scheme P (in the oracle model) that can invert an encryption of a random message with non-negligible advantage then there exists a (xed) deterministic straight-line (non-rewinding) extractor E with access to = (Enc, Rev, Dec_V) that can nd the witness for the underlying instance of the challenge ciphertext.

Suppose A is an adversary in the inversion game with success probability ϵ. Then the extractor E would works as follows: given a as input and acting as the challenger for adversary A, it chooses m $\$$ $\{0,1\}^k$ uniformly at random then runs A (1 , c) where c WEnc(a, m, 1) is the challenge. Queries issued by A are handled by E as follows:

> To answer any query Enc(x) asked by A, it forwards the query to the oracle and returns some answer c.
>
> To answer any query Rev(c) asked by A, it forwards the query to the oracle and returns some answer a.
>
> To answer any query $\text{Dec}_V(w, c)$ asked by A, the extractor rst issues a query Rev(c) to get some answer a. If $a \neq ?$, it would execute V (w, a), forwarding queries asked by V to similar to how it does for A. Finally, it forwards the query Dec(w, c) to to get some answer x. If a = ?, it returns ? to A otherwise it returns x.

While handling the queries made by A, if a decryption query $\text{Dec}_V(w, c)$ for the challenge ciphertext is issued by A, the extractor will pass this query to , and if the result of the decryption is $x \neq ?$ then the extractor will halt execution and output w as the witness for instance x. Otherwise, if after completing the execution of A, no such query was asked then the extractor outputs ?. We prove the following lemma.

Lemma 7.3.11. *For any PPT adversary A, instances a, if there exists a non-negligible function $\epsilon(.)$ such that:*

$$\Pr\left[A \ (1 \ , c) = m \mid m \ \$ \ \{0,1\}^k, c \quad \text{WEnc}(a, m, 1)\right] \quad \epsilon(\kappa) \tag{7.1}$$

Then there exists a PPT straight-line extractor E such that:

$$\Pr\left[E^{;A}(a) = w \wedge V \ (w, a) = 1\right] \quad \epsilon(\kappa) \quad \text{negl}(\kappa) \tag{7.2}$$

Proof. Let A be an adversary satisfying Equation (7.1) above and let AdvWin be the event that A succeeds in the inversion game. Furthermore, let ExtWin be the event that the extractor succeeds in extracting a witness (as in Equation (7.2) above). Observe that:

$$\begin{aligned}\Pr_{;m}[\text{ExtWin}] \quad & \Pr_{;m}[\text{ExtWin} \wedge \text{AdvWin}] \\ = 1 \quad & \Pr_{;m}[\overline{\text{ExtWin}} \vee \overline{\text{AdvWin}}] \\ = 1 \quad & \Pr_{;m}[\overline{\text{ExtWin}} \wedge \text{AdvWin}] \quad \Pr_{;m}[\overline{\text{AdvWin}}]\end{aligned}$$

Since Pr[AdvWin] ϵ for some non-negligible function ϵ, it su ces to show that Pr[$\overline{\text{ExtWin}}$ ^ AdvWin] is negligible. Note that, by our construction of extractor E, this event is equivalent to saying that the adversary succeeds in the inversion game but never asks a query of the form $\text{Dec}_V(w, c\,)$ for which the answer is x ~~6=~~ ? and so the extractor fails to recover the witness. For simplicity of notation de ne Win := $\overline{\text{ExtWin}}$ ^ AdvWin.

We will show that, with overwhelming probability over the choice of oracle , the probability of Win happening is negligible. That is, we will prove the following claim:

Claim 7.3.12. *For any negligible function* δ, Pr $\left[\Pr_m[\text{Win}] \quad \overline{p\ \delta}\right]$ negl(κ)

Proof. De ne Bad to be the event that A asks (directly or indirectly) a query of the form $\text{Dec}_V(w, c')$ for some $c' \neq c$ for which it has not asked Enc(x) = c previously. We have that:

$$\Pr_{;m}[\text{Win}] \quad \Pr_{;m}[\text{Win} \wedge \overline{\text{Bad}}] + \Pr_{;m}[\text{Bad}]$$

The probability of Bad over the randomness of is at most $1/2^n$ as it is the event that A hits an image of a sparse random injective function without asking the function on the preimage beforehand. Thus, Pr $_{;m}$[Bad] $1/2^n$.

It remains to show that Pr $_{;m}$[Win ^ $\overline{\text{Bad}}$] is also negligible. We list all possible queries that A could ask and argue that these queries do not help A in any way without also forcing the extractor to win as well. Speci cally, we show that for any such A that satis es the event (Win ^ $\overline{\text{Bad}}$), there exists another adversary $\widehat{A}$ that depends on A and also satis es the same event but does not ask any decryption queries (only encryption queries). This would then reduce to the standard case of inverting a random injective function, which is known to be hard. We de ne the adversary $\widehat{A}$ as follows. Upon executing A, it handles the queries issued by A as follows:

If A asks a query of the form Enc(x) then $\widehat{A}$ forwards the query to to get the answer.

If A asks a query of the form Rev(c) then since Bad does not happen, it must be the case that c = Enc(a, m) is an encryption that was previously asked by A and therefore $\widehat{A}$ returns a as the answer.

If A asks a query of the form Dec(w, c) then w must be a string for which V(w, a) = 0 or otherwise the extractor wins, which contradicts that $\overline{\text{ExtWin}}$ happens. If that is the case, since w is not a witness, $\widehat{A}$ would return *?* to A after running V (w, a) and answering its queries appropriately.

If A asks a query of the form Dec(w, c') for some $c' \neq c$ then, since Bad does not happen, it must be the case that A has asked a (direct or indirect) visible encryption query Enc(x') = c'. Therefore, $\widehat{A}$ would have observed this encryption query and can therefore run V (w, a') and return the appropriate answer (x or *?*) depending on the answer of V.

Given that $\widehat{A}$ perfectly emulates A's view, the only possibility that A could win the inversion game is by asking Enc(x) = c and hitting the challenge ciphertext, which is a negligible probability over the randomness of the oracle. By a standard averaging argument, we find that since $\Pr_{\$,m}[\text{Win} \wedge \overline{\text{Bad}}] \leq \delta(\kappa)$ for some negligible δ then $\Pr_{m}[\Pr_{\$}[\text{Win} \wedge \overline{\text{Bad}}] \leq \sqrt{\delta}] \geq 1 - \sqrt{\delta}$, which yields the result.

□

To conclude the proof of Lemma 7.3.11, we can see that the probability that the extractor wins is given by $\Pr[\text{ExtWin}] \geq 1 - \Pr[\overline{\text{ExtWin}} \wedge \text{AdvWin}] - \Pr[\overline{\text{AdvWin}}] \geq \epsilon(\kappa) - \text{negl}(\kappa)$ where ϵ is the non-negligible advantage of the adversary A.

□

It is clear that Construction 7.3.8 is a correct implementation. Furthermore, by Lemma 7.3.11, it satisfies the extractability property. Thus, this concludes the proof of Lemma 7.3.10. □

of Lemma 7.3.6. The existence of extractable instance-revealing witness encryption in the oracle model follows from Lemma 7.3.10 and Remark 7.3.9. □

7.3.4 Compiling out from IO

In this section, we show a simulatable compiler for compiling out . We adapt the approach outlined in Section 7.3.1 to the ideal IRWE oracle = (Enc, Rev, Dec_{V}) while making use of Lemma 7.2.4, which allows us to compile out in two phases: we first compile out part of to get an approximately-correct obfuscator $\widehat{O}^R$ in the random oracle model (that produces an obfuscation $\widehat{B}^R$ in the RO-model), and then use the previous result of [CKP15] to compile out the random oracle R and get an obfuscator O' in the plain-model. Since we are applying this lemma only a constant number of times (in fact, just twice), security should still be preserved. Specifically, we will prove the following claim:

Lemma 7.3.13. *Let $R \vee$ be a random oracle where "$\vee$" denotes a sub-model relationship (see Definition 7.2.1). Then the following holds:*

For any IO in the ideal model, there exists a simulatable compiler with correctness error $\epsilon < 1/200$ for it that outputs a new obfuscator in the random oracle R model.

[CKP15] For any IO in the random oracle R model, there exists a simulatable compiler with correctness error $\epsilon < 1/200$ for it that outputs a new obfuscator in the plain model.

Proof. The second part of Lemma 7.3.13 follows directly by [CKP15], and thus we focus on proving the first part of the claim. Before we start describing the compilation process, we present the following definition of canonical executions that is a property of algorithms in this ideal model and dependent on the oracle being removed.

Definition 7.3.14 (Canonical executions)**.** We define an oracle algorithm A relative to rIRWE to be in canonical form if before asking any $\mathsf{Dec}_{\mathsf{V}}(w, c)$ query, A would first get a Rev(c) then run V (w, a) on its own, making sure to answer any queries of V using . Furthermore, after asking a query $\mathsf{Dec}_{\mathsf{V}}(w, c)$ for which the returned answer is some message $m \neq \perp$, it would ask Enc(x) where $x = (a, m)$. Note that any oracle algorithm A can be easily modified into a canonical form by increasing its query complexity by at most a polynomial factor (since V is an extended poly-query algorithm).

Definition 7.3.15 (Query Types)**.** For any (not necessarily canonical) oracle algorithm A with access to a rIRWE oracle , we call the queries that are asked by A to as *direct* queries and those queries that are asked by V due to a call to Dec as *indirect* queries. Furthermore, we say that a query is *visible* to A if this query was issued by A and thus it knows the answer that is returned by . Conversely, we say a query is *hidden* from A if it is an indirect query that was not explicitly issued by A (for example, A would have asked a $\mathsf{Dec}_{\mathsf{V}}$ query which prompted V to ask its own queries and the answers returned to V will not be visible to A). Note that, once we canonicalize A, all indirect queries will be made visible since, by Definition 7.3.14, A will run V before asking $\mathsf{Dec}_{\mathsf{V}}$ queries and the query-answer pairs generated by V will be revealed to A.

We now proceed to present the construction of the random-oracle model obfuscator that, given an obfuscator in the model, would compile out and emulate queries to Dec and Rev while forwarding any Enc queries to R. Throughout this process, we assume that the obfuscators and the obfuscated circuits are all canonicalized according to Definition 7.3.14.

7.3.5 The new obfuscator $\widehat{\mathsf{IO}}^R$ in the random oracle model

Let $R = \{R_n\}_{n \in \mathbb{N}}$ be the (injective) random oracle where $R_n : \{0,1\}^n \to \{0,1\}^{2n}$. Given a δ-approximate obfuscator $\mathsf{IO} = (\mathsf{iO}, \mathsf{Ev})$ in the rIRWE oracle model, we construct an $(\delta + \epsilon)$-approximate obfuscator $\widehat{\mathsf{IO}} = (\widehat{\mathsf{iO}}, \widehat{\mathsf{Ev}})$ in the random oracle model.

Algorithm 1: EmulateCall

Input: Query-answer set Q, query q
Oracle: Random Oracle R
Output: ρ_q a query-answer pair containing the answer of query q
Begin:
if q is a query of type Enc(x) **then**
  Set ρ_q = $(x \mapsto R(x))_{\mathsf{Enc}}$
end
if q is a query of the form Rev(c) **then**
  if $\exists (x \mapsto c)_{\mathsf{Enc}} \in Q$ where $x = (a, m)$ **then**
    Set ρ_q = $(c \mapsto a)_{\mathsf{Rev}}$
  else
    Set ρ_q = $(c \mapsto ?)_{\mathsf{Rev}}$
  end
end
if q is a query of the form $\mathsf{Dec}_{\mathsf{V}}(w, c)$ **then**
  if $\exists (x \mapsto c)_{\mathsf{Enc}} \in Q$ **then**
    Initialize Q_{V} = ? and emulate $b \leftarrow \mathsf{V}(w, x)$
    for each query q_{V} asked by V **do**
      $\rho_{\mathsf{V}} \leftarrow \mathtt{EmulateCall}^R(Q \cup Q_{\mathsf{V}}, q_{\mathsf{V}})$
      $Q_{\mathsf{V}} = Q_{\mathsf{V}} \cup \rho_{\mathsf{V}}$
    end
    if $b = 1$ **then**
      Set ρ_q = $((w, c) \mapsto x)_{\mathsf{Dec}}$
    else
      Set ρ_q = $((w, c) \mapsto ?)_{\mathsf{Dec}}$
    end
  else
    Set ρ_q = $((w, c) \mapsto ?)_{\mathsf{Dec}}$
  end
end
Return ρ_q

Subroutine $\widehat{\mathsf{iO}}^R(C)$**:**

1. *Emulation phase*: Emulate $\mathsf{iO}(C)$. Let T_O be the transcript of this phase and initialize $Q_O := Q(T_O) = ?$. For every query q asked by $\mathsf{iO}(C)$, call $\rho_q \leftarrow \mathtt{EmulateCall}^R(Q_O, q)$ and add ρ_q to Q_O.

   Note that, since iO is a canonical algorithm, there are no hidden queries resulting from queries asked by V (via Dec queries) since we will always run V before asking/emulating a Dec query.

2. *Learning phase*: Set Q_B = ? to be the set of query-answer pairs learned during this phase. Set m = $2\ell_O/\epsilon$ where ℓ_O *j*iO*j* represents the number of queries asked by iO. Choose t $\overset{\$}{}$ [m] uniformly at random then for i = *f*1, ..., t*g*:

   Choose z_i $\overset{\$}{}$ *f*0, 1*g*^{jCj} uniformly at random

   Run Ev (B, z_i). For every query q asked by Ev (B, z_i), call and retrieve the answer ρ_q $\texttt{EmulateCall}^R(Q_O \,[\, Q_B, q)$ then add ρ_q to Q_B.

   Similar to Step 1, since Ev is a canonical algorithm and Enc is a injective function, with overwhelming probability, there will be no hidden queries as a result of asking any Dec queries.

3. The output of the RO model obfuscation algorithm $\widehat{\text{iO}}^R(C)$ will be $\widehat{B}$ = (B, Q_B).

Subroutine $\widehat{\text{Ev}}^R(\widehat{B}, z)$: To evaluate $\widehat{B}$ = (B, Q_B) on a new random input z we simply emulate Ev (B, z). For every query q asked by Ev (B, z), run and set ρ_q = $\texttt{EmulateCall}^R(Q_B, q)$ then add ρ_q to Q_B.

The running time of $\widehat{\text{iO}}$. We note that the running time of the new obfuscator $\widehat{\text{iO}}$ remains polynomial time since we are emulating the original obfuscation once followed by a polynomial number m of learning iterations. Furthermore, while we are indeed working with an oracle where the PPT V can have oracle gates to subroutines of , we emphasize that since V, which we are executing during EmulateCall, is a universal circuit evaluator, its e ective running time remains to be a strict polynomial in the size of V and so the issue of exponential or in nite recursive calls is non-existent.

Proving Approximate Correctness. Consider two separate experiments (real and ideal) that construct the random oracle model obfuscator exactly as described above but di er when evaluating $\widehat{B}$. Speci cally, in the real experiment, $\widehat{\text{Ev}}^R(\widehat{B}, z)$ emulates Ev (B, z) on a random input z and answers any queries by running Q_B, whereas in the ideal experiment, we execute $\widehat{\text{Ev}}^R(\widehat{B}, z)$ and answer the queries of Ev (B, z) using the actual oracle instead. In essence, in the real experiment, we can think of the execution as Ev$^{\widehat{}}(B, z)$ where $\widehat{}$ is the oracle simulated by using Q_B and oracle R. We will compare the real experiment with the ideal experiment and show that the statistical distance between these two executions is at most ϵ. In order to achieve this, we will identify the events that di erentiate between the executions Ev (B, z) and Ev$^{\widehat{}}(B, z)$.

Let q be a new query that is being asked by Ev$^{\widehat{}}(B, z)$ and handled by calling the subroutine $\texttt{EmulateCall}^R(Q_B, q)$. The following are the cases that should be handled:

1. If q is a query of type Enc(x), then the answer to q will be distributed the same in both experiments.

2. If q is a query of type Dec(w, c) or Rev(c) whose answer is determined by Q_B in the real experiment then it is also determined by $Q_O \cup Q_B \setminus Q_B$ in the ideal experiment and the answers are distributed the same.

3. If q is of type Dec(w, c) or Rev(c) that is not determined by $Q_O \cup Q_B$ in the ideal experiment then this means that we are attempting to decrypt a ciphertext for which we have not encrypted before and we will therefore answer it with $\perp$ with overwhelming probability. In that case, q will also not be determined by Q_B in the real experiment and we will answer it with $\perp$.

4. **Bad Event 1:** Suppose q is of type Dec(w, c) that is not determined by Q_B in the real experiment and yet is determined by $Q_O \cup Q_B$ in the ideal experiment to be some answer $x \neq \perp$. This implies that the query-answer pair $(x, c)_{\text{Enc}}$ is in $Q_O \cap Q_B$. That is, we are for the first time decrypting a ciphertext that was encrypted in Step 1 because we failed to learn the underlying x for ciphertext c during the learning phase of Step 2. In that case, in the real experiment, the answer would be $\perp$ since we do not know the corresponding message x whereas in the ideal experiment it would use the correct answer from $Q_O \cup Q_B$ and output x. However, we will show that this event is unlikely due to the learning procedure.

5. **Bad Event 2:** Suppose q is of type Rev(c) that is not determined by Q_B in the real experiment and yet is determined by $Q_O \cup Q_B$ in the ideal experiment. This implies that the query-answer pair $((a, m), c)_{\text{Enc}}$ is in $Q_O \cap Q_B$. That is, we are for the first time attempting to reveal the attribute of a ciphertext that was encrypted in Step 1 because we failed to learn the answer of this reveal query during the learning phase of Step 2. In that case, in the real experiment, the answer would be $\perp$ since we do not know the corresponding attribute a whereas in the ideal experiment it would use the correct answer from $Q_O \cup Q_B$ and output a. However, we will show that this event is unlikely due to the learning procedure.

For input x, let $E(x)$ be the event that Case 4 or 5 happen. Assuming that event $E(x)$ does not happen, both experiments will proceed identically the same and the output distributions of $\text{Ev}(B, x)$ and $\widehat{\text{Ev}}(B, x)$ will be statistically close. More formally, the probability of correctness for $\widehat{\text{IO}}$ is:

$$\begin{aligned}\Pr_x[\widehat{\text{Ev}}(B, x) \neq C(x)] = \Pr_x[\widehat{\text{Ev}}(B, x) \neq C(x) \wedge \neg E(x)] \\ + \Pr_x[\widehat{\text{Ev}}(B, x) \neq C(x) \wedge E(x)] \\ \Pr_x[\widehat{\text{Ev}}(B, x) \neq C(x) \wedge \neg E(x)] + \Pr_x[E(x)]\end{aligned}$$

By the approximate functionality of IO, we have that:

$$\Pr_x[\text{iO}(C)(x) \neq C(x)] = \Pr_x[\text{Ev}(B, x) \neq C(x)] \quad \delta(n)$$

Therefore,

$$\Pr_x[\widehat{\text{Ev}}(B,x) \neq C(x) \wedge :E(x)] = \Pr_x[\text{Ev}(B,x) \neq C(x) \wedge :E(x)] \quad \delta$$

We are thus left to show that $\Pr[E(x)] \quad \epsilon$. Since both experiments proceed the same up until E happens, the probability of E happening is the same in both worlds and we will thus choose to bound this bad event in the ideal world.

Claim 7.3.16. $\Pr_x[E(x)] \quad \epsilon$.

Proof. For all i 2 $[t]$, let $Q^{\emptyset}_{B_i} = Q_{B_i} \setminus Q_O$ be the set of query-answer pairs generated by the i'th evaluation $\text{Ev}(B, z_i)$ during the learning phase (Step 2) and are also generated during the obfuscation emulation phase (Step 1). In particular, $Q^{\emptyset}_{B_i}$ would contain the query-answer pairs $((a,m) \; ? \; c)_{\text{Enc}}$ for encryptions that were generated by the obfuscation and later discovered during the learning phase. Note that, since the maximum number of learning iterations $m > \ell_O$ and $Q^{\emptyset}_{B_i} \quad Q^{\emptyset}_{B_{i+1}}$, the number of learning iterations that would increase the size of the set of learned obfuscation queries is at most $2\ell_O$ since there are at most ℓ_O obfuscation ciphertexts that can be fully discovered during the learning phase and at most ℓ_O obfuscation ciphertexts that can be partially discovered (just nding out the underlying attribute a) via Rev queries during the learning phase.

We say $t \overset{\$}{} [m]$ is bad if it is the case that $Q^{\emptyset}_{B_t} \neq Q^{\emptyset}_{B_{t+1}}$ (i.e. t is an index of a learning iteration that increases the size of the learned obfuscation queries). This would imply that after t learning iterations in the ideal world, the nal evaluation $Q^{\emptyset}_{\widehat{B}} := Q^{\emptyset}_{B_{t+1}}$ would contain a new unlearned query-answer pair that was in Q_O. Thus, given that $m = 2\ell_O/\epsilon$, the probability (over the selection of t) that t is bad is at most $2\ell_O/m < \epsilon$. □

Proving Security. To show that the resulting obfuscator is secure, it su ces to show that the compilation process represented as the new obfuscator's construction is simulatable. We show a simulator S (with access to) that works as follows: given an obfuscated circuit B in the ideal model, it runs the learning procedure as shown in Step 2 of the new obfuscator $\widehat{\text{iO}}$ to learn the heavy queries Q_B then outputs $\widehat{B} = (B, Q_B)$. Note that this distribution is statistically close to the output of the real execution of $\widehat{\text{iO}}$ and, therefore, security follows. □

7.4 Separating IO from Instance Hiding Witness Encryption

In this section, we formally prove our second main separation theorem which states that there is no fully black-box construction of indistinguishability obfuscation from any *extended* predicate encryption scheme.

Theorem 7.4.1. *Assume that* NP *6* coAM *and that one-way functions exist. Then there exists no monolithic construction of indistinguishability obfuscation (IO) from predicate encryption (PE).*

In fact, we prove a stronger result by showing a separation of IO from a more generalized primitive, which we call extended *extractable instance-hiding* witness encryption, that implies extended predicate encryption and is a stronger variant of witness encryption.

Definition 7.4.2 (Extended Extractable Instance-Hiding Witness Encryption (ex-EIHWE))**.** Let V be a universal circuit-evaluator as de ned in De nition 6.5.9. For any given security parameter κ, an *extended extractable instance-hiding witness encryption* scheme consists of two PPT algorithms P = (Enc, Dec_V) de ned as follows:

Enc($a, m, 1$) : given an instance a 2 $f0,1g$, message m 2 $f0,1g$, and security parameter κ (and randomness as needed) it outputs c 2 $f0,1g$.

$\mathsf{Dec}_V(w, c)$: given ciphertext c and \witness" string w, it outputs a message m^0 2 $f0,1g$.

An extended extractable instance-hiding witness encryption scheme satis es the following completeness and security properties:

Correctness: For any security parameter κ, any m 2 $f0,1g$, and any $(w,(a,m))$ such that $V^P(w,(a,m))$ = 1, it holds that

$$\Pr_{\mathsf{Enc};\mathsf{Dec}}[\mathsf{Dec}(w, \mathsf{Enc}(a, m, 1\ )) = m] = 1$$

Instance-hiding Extractability: For any PPT adversary A and polynomial $p_1(.)$, there exists a PPT (black-box) straight-line extractor E and a polynomial function $p_2(.)$ such that the following holds. For security parameter κ, for all a_0, a_1 of the same length $ja_0j = ja_1j$ and any m_0 6= m_1 of the same length $jm_0j = jm_1j$, if:

$$\Pr\left[A(1^\kappa, c) = b \,j\, b \xleftarrow{\$} f0,1g, c \leftarrow \mathsf{Enc}(a_b, m_b, 1^\kappa)\right] \quad \frac{1}{2} + \frac{1}{p_1(\kappa)}$$

Then:

$$\Pr[E^A(a_0, a_1) = w \wedge 9\, b\ 2\ f0,1g\ s.t.\ V^P(w, a_b) = 1] \quad \frac{1}{p_2(\kappa)}$$

Generalizing the verification algorithm. Note that while the standard witness encryption scheme de nes the veri cation algorithm as operating on an witness-instance pair (w, a), since we are elevating the security to be instance-hiding, we can generalize the veri er to allow it to accept and act on not only the instance a but also the message m as input. That is, the veri er V in our de nition of instance-hiding witness encryption would accept (w, x) instead of just (w, a) where $x = (a, m)$. As a result, whenever it is unambiguous, we would often refer to x as the instance.

Given the above de nition of ex-IHWE, we prove the following theorem, which states that there is no fully black-box construction IO from extended IHWE.

Theorem 7.4.3. *Assume that* $\mathbf{NP}$ *6* $\mathbf{coAM}$ *and that one-way functions exist. Then there exists no* monolithic *construction of indistinguishability obfuscation from instance-hiding witness encryption.*

In Section 7.6.1, we prove that extended EIHWE implies extended predicate encryption. As a result, Theorem 7.4.1 would follow from Theorem 7.4.3, Lemma 7.6.1 and Lemma 6.6.3 (the transitivity lemma). Hence, for the remainder of this section we will focus on proving Theorem 7.4.3.

7.4.1 Overview of Proof Techniques

To prove Theorem 7.4.3, we will apply Lemma 7.2.7 for the idealized IHWE model (formally de ned in Section 7.4.2) to prove that there is no black-box construction of IO from any primitive P that can be oracle- xed constructed from . In particular, we will do so for P that is the extended IHWE primitive. Our task is thus twofold: (1) to prove that P can be oracle- xed constructed from and (2) to show a simulatable compilation procedure that compiles out from any IO construction. The rst task is proven in Section 7.4.3 and the second task is proven in Section 7.4.4. By Lemma 7.2.7, this would imply the separation result of IO from P and prove Theorem 7.4.3.

Our oracle, which is more formally de ned in Section 7.4.2, resembles an idealized version of a (instance-hiding) witness encryption scheme, which makes the construction of extended IHWE straightforward. As a result, the main challenge lies in showing a simulatable compilation procedure for IO that satis es De nition 7.2.2 in this idealized model, and therefore, it is instructive to look at how the compilation process works and what challenges are faced with dealing with oracle .

High-level Compiler Structure and Challenges

Recall that the solution adopted for the (extended) IRWE oracle was to canonicalize the obfuscated circuits such that queries made by the veri cation algorithm through decryption queries are made manifest so that they can be publicized without a ecting security. However, we shall see that this is not enough to argue that (approximate) correctness is maintained.

The Challenge Faced with (Instance-hiding) Witness Encryption. Here we are again faced with a problem when we try to emulate decryption queries. Very much like the setting of extended IRWE, since represents an idealized extended primitive, the veri cation algorithm V associated with the language de ned by might make oracle calls during the learning process. However, precisely due to the instance-hiding property, one cannot canonicalize the obfuscated circuits as before since we cannot e ciently extract the attribute or the message to run V (x, w) and discover the queries asked. This creates the possibility that the execution $B^{\emptyset}(x)$ might call such a hidden query and emulating it incorrectly.

Resolving the Challenge. While a full canonicalization is not possible here, we still perform it partially. Specifically, whenever the obfuscated circuit calls a decryption query $\mathsf{Dec}(w, c)$ and the underlying message x is successfully decrypted, the obfuscated circuit can now run $\mathsf{V}(x, w) = 1$ to reveal the queries asked. However, it is possible that the decryption fails, in which case we may have some hidden queries Q_{V} asked by an underlying $\mathsf{V}(x, w) = 0$ that cannot be revealed or learned. Nonetheless, we argue that the gathered information is enough for an approximately correct emulation of the plain-model obfuscated code.

7.4.2 The Ideal Model

In this section, we formally define the distribution of our ideal randomized oracle.

Definition 7.4.4 (Random Instance-hiding Witness Encryption Oracle)**.** Let V be a PPT universal circuit-evaluator as defined in Definition 2.1.1 that takes as input (w, x) where $x \in \{0,1\}^n$ and outputs $b \in \{0,1\}$. We define the following *random instance-hiding witness encryption* (rIHWE) oracle $\mathcal{O} = (\mathsf{Enc}, \mathsf{Dec}_{\mathsf{V}})$ as follows. We specify the sub-oracle $\mathcal{O}_n$ whose inputs are parameterized by n, and the actual oracle will be $\mathcal{O} = \{\mathcal{O}_n\}_{n \in \mathbb{N}}$.

> $\mathsf{Enc}: \{0,1\}^n \to \{0,1\}^{2n}$ is a random injective function. We will use the Enc oracle to encrypt a message $x \in \{0,1\}^n$ to obtain a ciphertext $c \in \{0,1\}^{2n}$.
>
> $\mathsf{Dec}_{\mathsf{V}}: \{0,1\}^s \to \{0,1\}^n \cup \{\bot\}$: Given $w \in \{0,1\}^k, c \in \{0,1\}^{2n}$ as inputs and $s = k + 2n$, $\mathsf{Dec}(w, c)$ allows us to decrypt the ciphertext c and get x as long as the predicate test is satisfied on (w, x). More formally, do as follow:
>
> 1. If $\nexists\, x$ such that $\mathsf{Enc}(x) = c$, output $\bot$. Otherwise, continue to the next step.
> 2. Find x such that $\mathsf{Enc}(x) = c$.
> 3. If $\mathsf{V}(w, x) = 0$ output $\bot$. Otherwise, output x.

We define a query-answer pair resulting from query q to subroutine $T \in \{\mathsf{Enc}, \mathsf{Dec}\}$ with some answer β as $(q \to \beta)_T$. The oracle $\mathcal{O}$ provides the subroutines for all inputs lengths but, for simplicity, and when n is clear from the context, we use $\mathcal{O} = (\mathsf{Enc}, \mathsf{Dec}_{\mathsf{V}})$ to refer to $\mathcal{O}_n$ for a fixed n.

7.4.3 (Instance-hiding) Witness Encryption exists relative to $\mathcal{O}$

In this section, we show how to construct a semantically-secure extended EIHWE with corresponding verification algorithm V relative to $\mathcal{O} = (\mathsf{Enc}, \mathsf{Dec}_{\mathsf{V}})$.

Construction 7.4.5 (Extended Extractable Instance-Hiding Witness Encryption)**.** Let V be a universal circuit-evaluator as defined in Definition 2.1.1. For any security parameter κ and oracle $\mathcal{O} = (\mathsf{Enc}, \mathsf{Dec}_{\mathsf{V}})$ sampled according to Definition 7.4.4, we will implement an extended EIHWE scheme P as follows:

WEnc$(a, m, 1^\kappa)$: Given $a \in \{0,1\}^*$, message $m \in \{0,1\}^*$ and security parameter 1^κ, let $n = 2\max(\kappa, |a|)$. Sample $r \xleftarrow{\$} \{0,1\}^{n^{n/2}-1}$ uniformly at random then output $c = \text{Enc}(x)$ where $x = (a, m)||r$.

WDec(w, c) : Given witness w and ciphertext c, let $x' = \text{Dec}_{\mathsf{V}}(w, c)$. If $x' \neq \perp$ then parse as $x' = (a', m')||r'$ and output m'. Otherwise, output $\perp$.

Lemma 7.4.6. *Construction 7.4.5 is a correct and subexponentially-secure oracle-fixed implementation (Definition 3.1.4) of extended extractable instance-hiding witness encryption in the ideal oracle model.*

To prove the security of this construction, we will show that if there exists an adversary A against scheme P (in the oracle model) that can distinguish between encryptions of two different messages with non-negligible advantage then there exists a (fixed) deterministic straight-line extractor E with access to $\mathcal{O} = (\text{Enc}, \text{Dec}_{\mathsf{V}})$ that can find the witness for the underlying instance of the challenge ciphertext.

Suppose A is an adversary in the indistinguishability game with success probability $1/2 + \epsilon$. Then the extractor E would works as follows: given (a_0, a_1) as input and acting as the challenger for adversary A, it runs $A^{\mathcal{O}}(1^\kappa, c)$ where $c \xleftarrow{\$} \text{WEnc}(a_b, m_b, 1^\kappa)$ is the challenge ciphertext. To answer any query Enc(x) asked by A, it forwards the query to the oracle $\mathcal{O}$ and returns some answer c. To answer any query Dec(w, c) for $c \neq c$ asked by A, the extractor will instead attempt to simulate this query for A instead of asking $\mathcal{O}$ directly. In particular, it first verifies whether there exists a query-answer pair $(x \stackrel{?}{=} c)_{\text{Enc}}$ that was generated by A previously. If such a query-answer pair exists then the extractor will run $\mathsf{V}^{\mathcal{O}}(w, x) = \beta$ and if $\beta = 1$ it returns to A the answer x. If such a query-answer pair does not exist or $\beta = 0$ then it returns $\perp$ to A. Note that, while running $\mathsf{V}^{\mathcal{O}}(w, x)$, the extractor will also need to answer the queries asked by V similar to how it did for A. While handling the queries made by A, if a decryption query $\text{Dec}_{\mathsf{V}}(w, c)$ for the challenge ciphertext is issued by A, the extractor will pass this query to $\mathcal{O}$, and if the result of the decryption is $x \neq \perp$ then the extractor will halt execution and output w as the witness for instance x. Otherwise, if after completing the execution of A, no such query was asked then the extractor outputs $\perp$. We prove the following lemma.

Lemma 7.4.7. *For any PPT adversary A, pair of instances a_0, a_1 of the same length $|a_0| = |a_1|$ and any $m_0 \neq m_1$ of the same length $|m_0| = |m_1|$, if there exists a non-negligible function $\epsilon(.)$ such that:*

$$\Pr\left[A^{\mathcal{O}}(1^\kappa, c) = b \mid b \xleftarrow{\$} \{0,1\}, c \xleftarrow{\$} \text{WEnc}(a_b, m_b, 1^\kappa)\right] \geq \frac{1}{2} + \epsilon(\kappa) \tag{7.3}$$

Then there exists a PPT extractor E such that:

$$\Pr\left[E^{\mathcal{O}, A}(a_0, a_1) = w \wedge \mathsf{V}^{\mathcal{O}}(w, a_b) = 1 \mid b \xleftarrow{\$} \{0,1\}\right] \geq \epsilon(\kappa) - \text{negl}(\kappa) \tag{7.4}$$

Let A be an adversary satisfying Equation (7.3) above and let AdvWin be the event that A succeeds in the distinguishing game. Furthermore, let ExtWin be the event that the extractor succeeds in extracting a witness (as in Equation (7.4) above). Observe that:

$$\Pr_{;b;r}[\overline{\text{ExtWin}}] \quad \Pr_{;b;r}[\overline{\text{ExtWin}} \wedge \text{AdvWin}] + \Pr_{;b;r}[\overline{\text{AdvWin}}]$$

Since Pr[AdvWin] $1/2 + \epsilon$ for some non-negligible function ϵ, it su ces to show that Pr[$\overline{\text{ExtWin}}$ ^ AdvWin] is negligibly close to 1/2. Note that, by our construction of extractor E, this event is equivalent to saying that the adversary succeeds in the distinguishing game but never asks a query of the form $\text{Dec}_V(w, c)$ for which the answer is $x \neq ?$ and so the extractor fails to recover the witness. For simplicity of notation de ne Win := $\overline{\text{ExtWin}}$ ^ AdvWin.

We will show that, with overwhelming probability over the choice of oracle , the probability of Win happening is only a negligible factor over the trivial advantage. That is, we will prove the following claim:

Claim 7.4.8. *For any negligible* ϵ*,* Pr [$\Pr_{b;r}$[Win] $1/2 + \epsilon$] negl(κ)

Proof. We rst show that we can construct an adversary $\widehat{A}$ that has the same winning advantage as A but does not ask decryption queries (only encryption queries). De ne Bad to be the event that A asks (directly or indirectly) a query of the form $\text{Dec}_V(w, c')$ for some $c' \neq c$ for which it has not asked Enc(x) = c' previously. Given the challenge c , the new adversary $\widehat{A}$ executes A (c) and answers the queries as follows: if the query is an encryption query then it is directly forwarded to to get the answer. If it is a Dec(w, c') query for some $c' \neq c$ then it will check if Enc(x') = c' was asked by A before in which case it runs V $(w, x') = \beta$ and returns the answer x' if and only if $\beta = 1$. If no such encryption was asked then event Bad has happened and we abort the execution. If a query Dec(w, c) was asked then we return $?$ immediately since $\overline{\text{ExtWin}}$ does not happen and therefore the Dec is expected to fail anyway. Thus, as long as Bad does not happen, the advantage of $\widehat{A}$ in winning the distinguishing game is at least the advantage of A.

It is straightforward to show, by a standard application of an averaging argument, that the expression Pr [$\Pr_{b;r}$[Bad] $\epsilon/3$] negl(κ). Speci cally, we know that the probability of Bad over the randomness of is at most $1/2^n$ as it is the event that A hits an image of a sparse random injective function without asking the function on the preimage beforehand. Thus, Pr $_{;b;r}$[Bad] $1/2^n$ and hence by an averaging argument Pr [$\Pr_{b;r}$[Bad] $\epsilon/3$] Pr $\left[\Pr_{b;r}[\text{Bad}] \quad 2^{n=2}\right]$ $2^{n=2}$.

We now proceed to show that $\widehat{A}$ only succeeds with negligible advantage while asking only encryption queries. De ne Hit to be the event that $\widehat{A}$ happens to ask Enc(x) = c .

Then we have that:

$$\Pr\left[\Pr_{b;r}[\text{Win}] \quad \frac{1}{2} + \epsilon\right] \quad \Pr\left[\Pr_{b;r}[\text{Win} \wedge \overline{\text{Bad}} \wedge \overline{\text{Hit}}] + \Pr_{b;r}[\text{Bad}] + \Pr_{b;r}[\text{Hit}] \quad \frac{1}{2} + \epsilon\right]$$

$$\Pr\left[\Pr_{b;r}[\text{Win} \wedge \overline{\text{Bad}} \wedge \overline{\text{Hit}}] \quad \frac{1}{2} + \frac{\epsilon}{3} - \Pr_{b;r}[\text{Bad}] \quad \frac{\epsilon}{3} - \Pr_{b;r}[\text{Hit}] \quad \frac{\epsilon}{3}\right]$$

$$\Pr\left[\Pr_{b;r}[\text{Win} \wedge \overline{\text{Bad}} \wedge \overline{\text{Hit}}] \quad \frac{1}{2} + \frac{\epsilon}{3}\right] + \Pr\left[\Pr_{b;r}[\text{Bad}] \quad \frac{\epsilon}{3}\right]$$

$$+ \Pr\left[\Pr_{b;r}[\text{Hit}] \quad \frac{\epsilon}{3}\right]$$

We can bound the event Hit from happening since it is the event that we invert the image of a random injective function. This can be done over the randomness of the oracle then, using an averaging argument, deduce that the probability that Hit happens for a non-negligible fraction of oracles is negligible.

We will thus focus on proving the first term. In doing so, we will reduce the problem of indistinguishability to that of predicting the output of a random Boolean function on a random point in the domain of this function. We then prove that the latter problem is hard using a compression technique which allows to argue that an adversary that can win the prediction game with non-negligible advantage can only do so for a negligible fraction of the oracles. We will make use of the following lemma that shows the existence of a randomized compression technique for random Boolean functions using adversaries against the prediction game.

Lemma 7.4.9 (Lemma 10.4 [DTT10]). *Let A be an oracle algorithm that makes q queries to some fixed oracle predicate $p : \{0,1\}^n \to \{0,1\}$, never queries the oracle on its input and satisfies the following for some ϵ:*

$$\Pr[A^p(x) = p(x)] \quad 1/2 + \epsilon$$

Then there exists a randomized encoder E and a randomized decoder D such that:

$$\Pr_r[D(E(p,r),r) = p] \quad \epsilon/q$$

and $|E(p,r)| \quad 2^n \quad \epsilon^2 2^n/q$.

For any fixed pair (a_0, a_1) and given adversary $\widehat{A}$ that wins in the indistinguishability game without asking any decryption queries, we can construct a new oracle-aided adversary $\widetilde{A}$ that aims to win in the experiment $\text{Exp}^P_{\widetilde{A}}(1^\lambda, a_0, a_1)$ as defined in Figure 7.2 without querying the oracle on its input and without asking any decryption queries. The adversary $\widetilde{A}$ will have access to oracle and P, which is defined as follows:

$$P(a_0, a_1, r, c_0, c_1) = \begin{cases} 0 \text{ if } c_0 = \text{Enc}(a_0 || 0 || r) \text{ and } c_1 = \text{Enc}(a_1 || 1 || r) \\ 1 \text{ if } c_0 = \text{Enc}(a_1 || 1 || r) \text{ and } c_1 = \text{Enc}(a_0 || 0 || r) \\ ? \text{ otherwise} \end{cases}$$

Experiment $\mathsf{Exp}^{P}_{A}(1^{\lambda}, a_0, a_1)$:

1. $r \xleftarrow{\$} \{0,1\}^{n/2-1}, b \xleftarrow{\$} \{0,1\}$
2. $c_0 \leftarrow \mathsf{Enc}(a_b \| b \| r), c_1 \leftarrow \mathsf{Enc}(a_{1-b} \| 1-b \| r)$
3. $b' \leftarrow A^{P,\mathcal{O}}(r, c_0, c_1)$ where A is not allowed to query P on (a_0, a_1, r, c_0, c_1) or (a_0, a_1, r, c_1, c_0)
4. Output 1 if $b = b'$ and 0 otherwise.

Figure 7.2: The 2-instance Exp^{P}_{A} Experiment

The adversary $\widetilde{A}$, given (r, c_0, c_1), would execute $b' \leftarrow \widehat{A}(c_0)$ and outputs b' as its answer. We can modify $\widetilde{A}$ so that whenever it issues a query to $\mathsf{Enc}(a \| b \| r) = c_b$, it would also call $\mathsf{Enc}(a \| 1-b \| r) = c_{1-b}$ followed by a call to $P(a, a, r, c_0, c_1)$. This ensures that any encryption query to $\mathcal{O}$ is translated into a query to P.

Note that P can be interpreted as an alternative description for $\mathcal{O}$. That is, given $P : \{0,1\}^{m} \to \{0,1\}$ where $m = 11n/2 - 1$, one can reconstruct $\mathcal{O}$ on any point in its domain. Define $P_{a_0,a_1} := P(a_0, a_1, ., ., .)$ to be the function P restricted to the attributes a_0 and a_1. By Lemma 7.4.9, we can encode P_{a_0,a_1} using at most $m - \epsilon^2 m/9q$ bits where q is the number of queries that A makes. Thus, we have compressed the entire oracle by saving $\alpha = \epsilon^2 m/9q$ bits. Hence, assuming that $\epsilon = \mathsf{negl}(\kappa)$ and $q = \mathsf{poly}(\kappa)$ we find that the fraction of oracles for which $\widetilde{A}$ can win on is at most $1/2 + \mathsf{negl}(\kappa)$.

□

Proof of Lemma 7.4.6. It is clear that the Construction 7.4.5 is correct. Furthermore, by Lemma 7.4.7, it also satisfies the extractability property. □

7.4.4 Compiling out $\mathcal{O}$ from IO

In this section, we show a simulatable compiler for compiling out $\mathcal{O}$. We formalize the approach outlined in Section 7.4.1 while making use of Lemma 7.2.4, which allows us to compile out $\mathcal{O}$ in two phases: we first compile out part of $\mathcal{O}$ to get an approximately-correct obfuscator $\widehat{\mathsf{IO}}^{R}$ in the random oracle model (that produces an obfuscation $\widehat{B}^{R}$ in the RO-model), and then use the previous result of [CKP15] to compile out the random oracle R and get an obfuscator IO' in the plain-model. Since we are applying this lemma only a constant number of times (in fact, just twice), security should still be preserved. Specifically, we will prove the following claim:

Lemma 7.4.10. *Let $R \vee \mathcal{O}$ be a random oracle where "$\vee$" denotes a sub-model relationship (see Definition 7.2.1). Then the following holds:*

For any IO in the ideal model, there exists a simulatable compiler for it with correctness error $\epsilon < 1/200$ that outputs a new obfuscator in the random oracle R model.

[CKP15] For any IO in the random oracle R model, there exists a simulatable compiler for it with error correctness $\epsilon < 1/200$ that outputs a new obfuscator in the plain model.

Proof. The second part of Lemma 7.4.10 follows directly from the previous result of [CKP15], and thus we focus on proving the first part of the claim. Before we start describing the compilation process, we present the following definition of canonical executions that is a property of algorithms in this ideal model and dependent on the oracle being removed, then we describe the notation and terminology used throughout the proof.

Definition 7.4.11 (Canonical executions)**.** We define an oracle algorithm A relative to rIHWE to be in canonical form if right after asking any $\mathsf{Dec}_{\mathsf{V}}(w, c) = x$ where $x \neq \perp$, A would also additionally ask the query $\mathsf{Enc}(x)$ followed by directly executing $\mathsf{V}(w, x)$ on its own. Note that any algorithm A can be easily modified into a canonical form by increasing its query complexity at most by a polynomial factor (since V is a PPT algorithm).

Definition 7.4.12 (Query Types)**.** For any canonical oracle algorithm A with access to a rIHWE oracle , we call the queries that are asked by A to as *direct* queries and those queries that are asked by V due to a call to Dec as *indirect* queries. Furthermore, we say that a query is *visible* to A if this query was issued by A and thus it knows the answer that is returned by . Conversely, we say a query is *hidden* from A if it is an indirect query that was not explicitly issued by A (for example, A would have asked a $\mathsf{Dec}_{\mathsf{V}}$ query which prompted V to ask its own queries and the answers returned to V will not be visible to A).

Note that, while all direct queries are visible to A, an indirect query q may or not may visible to A; an indirect query q is visible to A if it runs V and q is asked by V then forwarded to A (where q was not previously asked by A in an explicit manner).

Transcripts. We use bold fonts (e.g., $\mathbf{T}$) to denote the distribution from which we sample an actual transcript T $\mathbf{T}$. For any transcript T of some oracle algorithm, we define $U(T)$ to be the set of query-answer pairs asked of during the generation of T and define $Q(T)$ to be the projection of $U(T)$ onto the first element, which represents the set of queries only.

We now proceed to present the construction of the random-oracle model obfuscator that, given an indistinguishability obfuscator $\mathsf{IO} = (\mathsf{iO}, \mathsf{Ev})$ in the model, would compile out and emulate queries to Dec while forwarding any Enc queries to R. Throughout this process, we assume, without loss of generality, that the ideal-model obfuscator iO and the ideal-model evaluation of the obfuscation Ev are both in canonical form according to Definition 7.4.11. For these canonical algorithms, let $\ell_O, \ell_B = \mathsf{poly}(n)$, respectively, be the number of queries asked by the ideal-model obfuscator iO and the evaluation Ev to the rIHWE oracle .

Algorithm 2: `EmulateCall`

Input: Query-answer set U and query q
Oracle: Random oracle R
Output: (ρ_q, W_H) where ρ_q is a query-answer pair containing the answer of query q and W_H is the set of indirect query-answer pairs (those asked by V)
Begin:
Initialize W_H = ?
if q is a query of type Enc(x) for any $x \in \{0,1\}^n$ **then**
  Set $\rho_q = (x \mapsto R(x))_{\mathsf{Enc}}$
end
if q is a query of the form $\mathsf{Dec}_{\mathsf{V}}(w, c)$ for any $w \in \{0,1\}$ and $c \in \{0,1\}^{2n}$ **then**
  if $(x \mapsto c)_{\mathsf{Enc}} \in U$ **then**
    Emulate $b \leftarrow \mathsf{V}(w, x)$
    for each query q_{V} asked by V **do**
      $(\rho_{\mathsf{V}}, W_{\mathsf{V}}) \leftarrow \mathtt{EmulateCall}^R(U \cup W_H, q_{\mathsf{V}})$
      $W_H = W_H \cup (\rho_{\mathsf{V}} \cup W_{\mathsf{V}})$
    end
    if $b = 1$ **then**
      Set $\rho_q = ((w, c) \mapsto x)_{\mathsf{Dec}}$ /* Canonical caller can now call $\mathsf{V}(w, x)$ to publicize W_H */
    else
      Set $\rho_q = ((w, c) \mapsto ?)_{\mathsf{Dec}}$
    end
  else
    Set $\rho_q = ((w, c) \mapsto ?)_{\mathsf{Dec}}$
  end
end
Return (ρ_q, W_H)

The new obfuscator $\widehat{\mathsf{iO}}^R$ in the random oracle model

Let $R = \{R_n\}_{n \in \mathbb{N}}$ be the (injective) random oracle where $R_n : \{0,1\}^n \to \{0,1\}^{2n}$. Given a δ-approximate obfuscator $\mathsf{iO} = (\mathsf{iO}, \mathsf{Ev})$ in the rIHWE oracle model, we construct an $(\delta + \epsilon)$-approximate obfuscator $\widehat{\mathsf{iO}} = (\widehat{\mathsf{iO}}, \widehat{\mathsf{Ev}})$ in the random oracle model.

Subroutine $\widehat{\mathsf{iO}}^R(C)$:

1. *Emulation phase*: Given input circuit $C \in \mathcal{C}_n$, emulate $\mathsf{iO}(C)$ and let B be the output of this emulation. Let T_O be the transcript of this phase and initialize $Q_O := U(T_O) = ?$. For every query q asked by $\mathsf{iO}(C)$:

   Call $(\rho_q, W_H) \leftarrow \mathtt{EmulateCall}^R(Q_O, q)$ as shown in Algorithm 2 and return the answer in ρ_q

Add ρ_q and W_H to Q_O

Note that we will not pass *any* query-answer pairs Q_O asked by iO to the output of this new obfuscation. All of the query-answer pairs in Q_O, be it visible or hidden, will only be used for emulation purposes in the following step.

2. *Learning phase*: Let Q_B be the set of all visible queries that will be learned, Q_B^h be the set of hidden queries that we will keep track of for emulation purposes. Set $m = 3\ell_O/\epsilon$ where ℓ_O jiOj represents the number of queries asked by iO. Choose $t \xleftarrow{\$} [m]$ uniformly at random then for $i = \{1, ..., t\}$ do the following:

   Choose $z_i \xleftarrow{\$} \{0,1\}^{|C|}$ uniformly at random

   Run Ev (B, z_i). For every query q asked by Ev (B, z_i):

   - Call (ρ_q, W_H) $\texttt{EmulateCall}^R(Q_O \cup Q_B \cup Q_B^h, q)$ and return the answer in ρ_q
   - Add ρ_q to Q_B and add W_H to Q_B^h.

   Since Ev is a canonical algorithm, in addition to the direct queries asked by Ev (B, z_i) to , the resulting query set Q_B would also include the indirect queries that have resulted from Ev (B, z_i) running V $(w, x) = 1$ on a successful decryption query.

3. The output of the random oracle model obfuscation algorithm $\widehat{\text{iO}}^R(C)$ will be $\widehat{B} = (B, Q_B)$ where Q_B is the set of visible (direct and/or indirect) query-answer pairs.

Subroutine $\widehat{\text{Ev}}^R(\widehat{B}, z)$**:** Initialize $Q_{\widehat{B}} = ?$ to be the set of queries asked when evaluating $\widehat{B}$. To evaluate $\widehat{B} = (B, Q_B)$ on a new random input z we simply emulate Ev (B, z). For every query q asked by Ev (B, z), run and set $(\rho_q, W_H) = \texttt{EmulateCall}^R(Q_B \cup Q_{\widehat{B}}, q)$ then return the answer in ρ_q and add (ρ_q, W_H) to $Q_{\widehat{B}}$.

Note that, since we are emulating query calls with respect to $Q_B \cup Q_{\widehat{B}}$, all the indirect query-answer pairs W_H returned from queries of the form $\text{Dec}_V(w, c)$ are due to knowing the underlying $(x \| c)_{\text{Enc}}$ (i.e. $(x \| c)_{\text{Enc}} \in Q_B \cup Q_{\widehat{B}}$) so W_H is visible and therefore used in the emulation of any subsequent queries issued during this execution.

The running time of $\widehat{\text{iO}}$**.** We note that the running time of the new obfuscator $\widehat{\text{iO}}$ remains polynomial time since we are emulating the original obfuscation once followed by a polynomial number m of learning iterations. Furthermore, while we are indeed working with an oracle where the PPT universal circuit evaluator V can have oracle gates to subroutines of , we emphasize that in our framework of extended primitives, the e ective running time of V, which we are executing during `EmulateCall`, remains to be a strict polynomial in the size of V and so the issue of exponential or in nite recursive calls is non-existent.

Proving Approximate Correctness. Let Q_S = (Q_O [Q_B [Q_B^h [$Q_{\widehat{B}}$) be the set of all query-answer pairs asked during the emulation, learning, and evaluation phases. We de ne the following hybrid experiments to show that the correctness of the new random-oracle model obfuscator that simulates an execution of a -model obfuscation is su ciently close to the correctness of an ideal execution of the -model obfuscation. We denote the distribution of Hybrid i as H^i = ($Q_O^i, Q_B^i, Q_B^{i;h}, Q_{\widehat{B}}^i$).

Hybrid 0: This is the real experiment, which is represented as the construction in Section 7.4.4.

Hybrid 1: This is the same as Hybrid 0 except for two di erences:

- The queries asked during emulation (Step 1) and learning (Step 2) answered relative to a true oracle instead of emulating them using `EmulateCall`.
- We modify the nal execution $\widehat{\text{Ev}}(\widehat{B}, z)$ such that, when we execute Ev (B, z), we would forward all queries to instead of using `EmulateCall` *except* when the query is of the form Dec(w, c) for some (x 7! c)$_{\text{Enc}}$ 2 Q_B [$Q_{\widehat{B}}$, in which case we *do not* forward this query to and use *?* as the answer instead. We will refer to such decryption queries as being *unknown*.

Hybrid 2: This is the ideal experiment, where *all* the queries asked during emulation, learning and nal execution are answered relative to a true oracle . That is, all queries whose answers are emulated using `EmulateCall` throughout the obfuscation and nal evaluation phases will instead be answered directly using .

Claim 7.4.13. (H^0, H^1) negl(κ)

Proof. We rst observe that the emulation and learning processes of Hybrid 0 is a statistical simulation of the true oracle. Furthermore, if the nal execution phase asks a query Dec(w, c) for which (x 7! c)$_{\text{Enc}}$ 2 Q_B [$Q_{\widehat{B}}$ then the answer in both hybrids will be consistent with each other otherwise the answer is *?* in both hybrids. Consequently, any encryption query will be answered in Hybrid 0 using random oracle R and in Hybrid 1 using , and Hybrid 0 statistically simulates decryptions queries without access to .

The only event that can cause a discrepancy between these two hybrids is during the emulation and learning phases where a decryption query Dec$_{\mathsf{V}}$(w, c) is issued and (x 7! c)$_{\text{Enc}}$ 2 Q_S but c is a valid ciphertext (i.e. there exists x such that Enc(x) = c). In that case, in Hybrid 0, the answer to such a query will be *?* whereas the answer in Hybrid 1 will be x. However, the probability of that happening in Hybrid 1 (over the randomness of oracle) is at most (2^n p)/(2^{2n} p) $1/2^{n-1}$ = negl(κ) given that $p = tl_B l_O$ = poly(κ). □

Claim 7.4.14. (H^1, H^2) ϵ

Proof. Note that, since the emulation and learning processes are the same in both hybrids, we have that ($Q_O^1, Q_B^1, Q_B^{1;h}$) = ($Q_O^2, Q_B^2, Q_B^{2;h}$) and so we omit the superscripts when referencing these query-answer sets for simplicity of notation. In Hybrid 1, the execution of $\widehat{B}$

emulates Ev (B, z) on a random input z and answers all queries using except for unknown decryption queries, whereas in Hybrid 2, we execute $\widehat{B}$ and answer *all* of Ev (B, z)'s queries using the actual oracle . In essence, in Hybrid 1, we can think of the execution as Ev$^{\widehat{}}(B, z)$ where $\widehat{}$ is the oracle simulated by $\widehat{B}$ using Q_B and . We will identify the events that di erentiate between the executions Ev (B, z) and Ev$^{\widehat{}}(B, z)$.

Assume that the query-answer pairs so far during the execution of $\widehat{B}$ are the same in both hybrids. That is, we start with some $Q^1_{\widehat{B}} = Q^2_{\widehat{B}} = Q_{\widehat{B}}$ for the sake of proving inductively that any subsequent query-answer pairs are closely distributed between the two hybrids. Let q be a new (possibly indirect) query that is being asked by Ev$^{\widehat{}}(B, z)$. We present a case-by-case analysis of all possible query types to identify the cases that can cause discrepancies between the two hybrids:

1. If q is a query of type Enc(x), then it will be answered the same in both hybrids using the true oracle .

2. If q is of type Dec(w, c) for which there exists $(x \; ? \; c)_{\text{Enc}} \in Q_B \cup Q_{\widehat{B}}$ then the answer to q would be determined the same in both hybrids. In particular, both hybrids will run V $(w, c) = b$, adding any indirect queries to $Q_{\widehat{B}}$ and answering q with x if and only if $b = 1$.

3. If q is of type Dec(w, c) such that $(x \; ? \; c)_{\text{Enc}} \notin Q_S$ then this means that we are attempting to decrypt a ciphertext for which we have not encrypted before (either directly or indirectly). In Hybrid 2, since Enc is a sparse random injective function, the answer would be $?$ with overwhelming probability as c would be invalid, and in Hybrid 1 the answer will also be $?$ by the de nition of $\widehat{\text{Ev}}(\widehat{B}, z)$ in this hybrid. Note that in both hybrids, there will not be any indirect queries in $Q_{\widehat{B}}$ as a result of this query since the ciphertext c is deemed invalid and V will not even be executed.

4. Suppose q is of type Dec(w, c) that is not determined by $Q_B \cup Q_{\widehat{B}}$ in Hybrid 1 and yet is determined by Q_S in Hybrid 2. We list the di erent cases here that may cause problems:

   (a) If q is of type Dec(w, c) such that $(x \; ? \; c)_{\text{Enc}} \in Q_S \cap (Q_B \cup Q_{\widehat{B}})$ and V $(w, x) = 0$ then this means that we are attempting to decrypt a ciphertext for which the decryption will fail. The answer will be $?$ in Hybrid 1 by the de nition of $\widehat{\text{Ev}}(\widehat{B}, z)$ and is also $?$ in Hybrid 2 since we are using the real oracle . However, one crucial point here is that, while there will possibly exist hidden queries due to the internal execution of V (w, x) in Hybrid 2, such hidden queries will *not* be present in Hybrid 1 as we do not run V there. As we shall see later on, this presents a potential source of inconsistency for subsequent queries.

   (b) **Bad Event 1:** The query-answer pair $(x \; ? \; c)_{\text{Enc}}$ is in $Q_O \cap (Q_B \cup Q_{\widehat{B}})$ and V $(w, x) = 1$. That is, we are for the rst time decrypting a ciphertext that was encrypted in Step 1 because we failed to learn the answer of this decryption query

during the learning phase of Step 2. In that case, in Hybrid 1, the answer would be $\perp$ since we do not know the corresponding message x whereas in Hybrid 2 it would use the answer consistent with $Q_O \cup Q_S$ and output x.

(c) **Bad Event 2:** The query-answer pair $(x \| c)_{\mathsf{Enc}}$ is in $Q_B^h \setminus (Q_B \cup Q_{\widehat{B}})$ and $V(w, x) = 1$. That is, we are for the first time decrypting a ciphertext from a hidden encryption query in Step 2. In that case, in Hybrid 1, the answer would be $\perp$ since we do not know the corresponding message x whereas in Hybrid 2 it would use the answer consistent with $Q_B^h \cup Q_S$ and output x.

(d) **Bad Event 3:** The query-answer pair $(x \| c)_{\mathsf{Enc}}$ is in $Q_{\widehat{B}}^h \setminus (Q_B \cup Q_{\widehat{B}})$ and $V(w, x) = 1$ where $Q_{\widehat{B}}^h$ is the set of hidden queries generated in Hybrid 2 as a result of queries from Case 4a. In that case, in Hybrid 1, the answer would be $\perp$ since we do not know the corresponding message x whereas in Hybrid 2 it would use the answer consistent with $Q_{\widehat{B}}^h \cup Q_S$ and output x.

While the above bad events will generate indirect queries in Hybrid 2 (the ciphertext is valid there), these indirect queries will not be present in Hybrid 1, and may cause inconsistencies between the two hybrids when evaluating subsequent queries. However, as long as we show that the bad events happen with small probability, those indirect queries will also be generated in Hybrid 2 only with small probability.

For input x, let $E_1(x)$ be the event that Case 4b happens, $E_2(x)$ be the event that Case 4c happens, and $E_3(x)$ be the event that Case 4d happens. Assuming that event $E(x) = (E_1(x) \vee E_2(x) \vee E_3(x))$ does not happen, both experiments will proceed identically the same and the output distributions of $\mathsf{Ev}(B, x)$ and $\widehat{\mathsf{Ev}}(B, x)$ will be statistically close. More formally, the probability of correctness for $\widehat{\mathsf{IO}}$ is:

$$\begin{aligned}\Pr_x[\widehat{\mathsf{Ev}}(B, x) \not\perp C(x)] &= \Pr_x[\widehat{\mathsf{Ev}}(B, x) \not\perp C(x) \wedge \neg E(x)] \\ &\quad + \Pr_x[\widehat{\mathsf{Ev}}(B, x) \not\perp C(x) \wedge E(x)] \\ &\leq \Pr_x[\widehat{\mathsf{Ev}}(B, x) \not\perp C(x) \wedge \neg E(x)] + \Pr_x[E(x)]\end{aligned}$$

By the approximate functionality of IO, we have that:

$$\Pr_x[\mathsf{iO}(C)(x) \not\perp C(x)] \leq \Pr_x[\mathsf{Ev}(B, x) \not\perp C(x)] + \delta(n)$$

Therefore,

$$\Pr_x[\widehat{\mathsf{Ev}}(B, x) \not\perp C(x) \wedge \neg E(x)] \leq \Pr_x[\mathsf{Ev}(B, x) \not\perp C(x) \wedge \neg E(x)] + \delta$$

.

We are thus left to show that $\Pr[E(x)] \leq \Pr[E_1(x)] + \Pr[E_2(x) \mid \overline{E}_1(x)] + \Pr[E_3(x) \mid \overline{E}_1(x) \wedge \overline{E}_2(x)] + \epsilon$. Since both experiments proceed the same up until E happens, the probability of E happening is the same in both worlds and we will thus choose to bound these bad events in Hybrid 2.

Claim 7.4.15. $\Pr_x[E_1(x)] \quad \epsilon/3$.

Proof. Recall that E_1 is the event that during the final execution of the obfuscation $\widehat{B}$ on a random input, a Dec(w, c) query is asked for some c that was generated during the obfuscation phase but such a decryption query was never asked during the learning phase (Case 4b). For all i 2 $[t]$, let $Q^{\emptyset}_{B_i}$ = $(Q_{B_i}$ [$Q^{h}_{B_i})$ \ Q_O be the set of query-answer pairs generated by the i'th evaluation Ev (B, z_i) during the learning phase (Step 2) and are also generated during the obfuscation emulation phase (Step 1). Note that, since the maximum number of learning iterations $m > \ell_O$ and $Q^{\emptyset}_{B_i} \quad Q^{\emptyset}_{B_{i+1}}$, the number of learning iterations that increase the size of the set of learned obfuscation queries is at most ℓ_O. We say t $^{\$}$ $[m]$ is bad if it is the case that $Q^{\emptyset}_{B_t}$ ~~6~~ $Q^{\emptyset}_{B_{t+1}}$ (i.e. t is an index of a learning iteration that increases the size of the learned obfuscation queries). This would imply that after t learning iterations in Hybrid 1, the real execution $Q^{\emptyset}_{\widehat{B}}$:= $Q^{\emptyset}_{B_{t+1}}$ would contain a new unlearned query that was in Q_O. Thus, given that m = $3\ell_O/\epsilon$, the probability (over the selection of t) that t is bad is at most $\ell_O/m < \epsilon/3$. □

Claim 7.4.16. $\Pr_x[E_2(x)$ *j* $\overline{E}_1(x)] \quad \text{negl}(\kappa)$

Proof. Recall that E_2 is the event that during the final execution of the obfuscation $\widehat{B}$ on a random input, a query is asked that was issued during the learning phase but was part of the hidden set and therefore never publicized (Case 4c).

For all i 2 $[t]$, let Q_{B_i} = $(Q_{B_i}$ [$Q^{h}_{B_i})$ be the set of (public and hidden) query-answer pairs generated by the i'th evaluation Ev (B, z_i) during the learning phase (Step 2) and $Q_{\widehat{B}}$ be all the query-answer pairs of the final evaluation of the obfuscation. Note that in Hybrid 2, the real oracle is used and therefore each execution of Ev (B, z_i) is independent of the other executions. In particular, we can think of the final evaluation as iteration t + 1 of the learning phase with query-answer pairs $Q_{B_{t+1}}$ = $Q_{\widehat{B}}$. Now, we can think of an intermediate hybrid with distribution $H^{2'}$ where the executions are randomly permuted in a way such that execution t + 1 in Hybrid 2 is now the first execution in Hybrid $2^{\emptyset}$. Note that, since the executions are independent, the distributions H^2 and $H^{2'}$ are equivalent and it remains to show that E_2 is unlikely to happen in Hybrid $2^{\emptyset}$. Specifically, we need to show that the first execution of the learning phase (with query-answer pairs $Q_{B_{t+1}}$) does not decrypt a ciphertext generated by any previous hidden queries in the learning phase. However, since this is the first execution, there exists no hidden queries from previous executions and it suffices to argue that it does not decrypt a ciphertext generated by any hidden queries $Q^{h}_{B_{t+1}}$ from its own execution.

Since E_1 does not happen and since the probability that Ev (B, z_{t+1}) decrypts a ciphertext that was never encrypted during the whole process is negligible, the only ciphertexts that this execution will attempt to decrypt are ones that were generated during this very same execution. Therefore, E_2 only happens with negligible probability when a ciphertext that was never encrypted before is being decrypted.

□

Claim 7.4.17. $\Pr[E_3(x)$ *j* $\overline{E}_1(x)$ ^ $\overline{E}_2(x)] \quad \text{negl}(\kappa)$

Proof. We bound this event in Hybrid 1. Given that the nal execution in this hybrid does not issue unknown decryption queries, there are no hidden queries generated in this hybrid and so E_3 only happens with negligible probability when a ciphertext that was never encrypted before is being decrypted. □

Combining the results of the above claims, we get that Pr$_x$[$E(x)$] Pr[$E_1(x)$] + Pr[$E_2(x)$ *j* $\overline{E}_1(x)$] + Pr[$E_3(x)$ *j* $\overline{E}_1(x)$ ^ $\overline{E}_2(x)$] $\epsilon/3$ + negl(n) ϵ.

□

Proving Security. To show that the resulting obfuscator is secure, it su ces to show that the compilation process represented as the new obfuscator's construction is simulatable. We show a simulator S (with access to) that works as follows: given an obfuscated circuit B in the ideal model, it runs the learning procedure as shown in Step 2 of the new obfuscator $\widehat{\text{IO}}$ to learn the heavy queries Q_B then outputs $\widehat{B}$ = (B, Q_B). Note that this distribution is statistically close to the output of the real execution of $\widehat{\text{IO}}$ and, therefore, security follows.

□

7.5 Separating IO from Homomorphic Witness Encryption

In this section, we formally prove our third main separation theorem which states that there is no fully black-box construction of indistinguishability obfuscation from any *extended* fully homomorphic encryption scheme.

Theorem 7.5.1. *Assume the existence of one-way functions and that* $\mathbf{NP} \not\subseteq \mathbf{coAM}$. *Then there exists no monolithic construction of indistinguishability obfuscation (IO) from fully homomorphic encryption (FHE).*

In fact, we prove a stronger result by showing a separation of IO from a more generalized and powerful version of witness encryption, which we call extended *extractable homomorphic* witness encryption. In essence, this is an instance-revealing witness encryption (De nition 2.6.2) with added homomorphic capabilities and extractable security.

Definition 7.5.2 (Extended Extractable Instance-Revealing Homomorphic Witness Encryption)**.** Let V be a universal circuit-evaluator that takes instance a and witness w and either accepts or rejects. Furthermore, let F be a universal circuit evaluator that takes as input a sequence of messages $m_1, ..., m_k$ to output some value m'. For any given security parameter κ, an extended *extractable instance-revealing homomorphic witness encryption* (ex-EIRHWE) scheme de ned for (V, F) consists of four PPT algorithms P = (Enc, Rev, Dec$_V$, Eval$_F$) de- ned as follows:

> Enc($a, m, 1$) : given an instance a *2* $f0,1g$ and a message m *2* $f0,1g$, and security parameter κ (and randomness as needed) it outputs c *2* $f0,1g$.

Rev(c) : given ciphertext c outputs $a \in \{0,1\} \cup \{?\}$.

$\text{Dec}_V(w, c)$: given ciphertext c and "witness" string w, it outputs a message $m' \in \{0,1\}$.

$\text{Eval}_F(c_1, ..., c_k)$: given ciphertexts[5] $c_1 = \text{Enc}(a_1, m_1, 1^\lambda), ..., c_k = \text{Enc}(a_k, m_k, 1^\lambda)$, it outputs $\{c'_i\}_{i=1}^{p}$ where $p = |U|$ and $U \subseteq A$ is the set of distinct attributes from $A = \{a_1, ..., a_k\}$.

An extended extractable instance-revealing homomorphic witness encryption scheme satisfies the following completeness and security properties:

Decryption Correctness: For any security parameter κ, we have that the following two conditions are satisfied:

- For any (w, a) such that $V^P(w, a) = 1$, and any m:

$$\Pr_{\text{Enc,Dec}}[\text{Dec}(w, \text{Enc}(a, m, 1^\lambda)) = m] = 1$$

- For any (w_i, a_i) such that $V^P(w_i, a_i) = 1$, and any $m_1, ..., m_k$:

$$\Pr_{\text{Enc,Dec,Eval}}[\text{Dec}(w_i, \text{Eval}(c_1, ..., c_k)) = F^P(m_1, ..., m_k)|_i] = 1$$

  where $c_i = \text{Enc}(a_k, m_k, 1^\lambda)$ for all $i \in [k]$ and $F^P(m_1, ..., m_k)|_i$ denotes the ith secret share of $F^P(m_1, ..., m_k)$.

Instance-Revealing Correctness: For any security parameter κ and any (a, m) it holds that:

$$\Pr_{\text{Enc,Rev}}[\text{Rev}(\text{Enc}(a, m, 1^\lambda)) = a] = 1$$

Furthermore, for any c for which there is no a, m, κ such that $\text{Enc}(a, m, 1^\lambda) = c$ it holds that $\text{Rev}(c) = ?$.

Extractability: For any PPT adversary A and polynomial $p_1(.)$, there exists a PPT (black-box) straight-line extractor E and a polynomial function $p_2(.)$ such that the following holds. For any security parameter κ, for all a and any $m_0 \neq m_1$ of the same length $|m_0| = |m_1|$, if:

$$\Pr\left[A(1^\kappa, c) = b \mid b \xleftarrow{\$} \{0,1\}, c \leftarrow \text{Enc}(a, m_b, 1^\lambda)\right] \geq \frac{1}{2} + \frac{1}{p_1(\kappa)}$$

Then:

$$\Pr[E^A(a) = w \wedge V^P(w, a) = 1] \geq \frac{1}{p_2(\kappa)}$$

[5] Without loss of generality, we exclude from the input to the evaluation any explicit in-the-clear representation of a function $f \in F$ to compute on the underlying messages as we can interpret the function f as a "message" $m_f = f$, encrypt it to get c_f and include it as input to the evaluation.

Given the above de nition of ex-EIRHWE, we prove the following theorem, which states that there is no fully black-box construction IO from extended EIRHWE.

Theorem 7.5.3. *Assume the existence of one-way functions and that* $\mathbf{NP} \not\subset \mathbf{coAM}$*. Then there exists no* monolithic *construction of indistinguishability obfuscation from extractable instance-revealing homomorphic witness encryption for any PPT veri cation algorithm* V *and function* F*.*

Since extended EIRHWE implies extended fully homomorphic encryption (in fact, in Sections 7.6.2 and 7.6.3, we show that it also implies other more powerful primitives such as extended attribute-based FHE and spooky encryption), Theorem 7.5.1 follows from Theorem 7.5.3, the above observations, and Lemma 6.6.3 (the transitivity lemma). As a result, for the remainder of this section we will focus on proving Theorem 7.5.3.

7.5.1 Overview of Proof Techniques

To prove Theorem 7.5.3, we will apply Lemma 7.2.7 for the rIRHWE model (formally de ned in Section 7.5.2) to prove that there is no black-box construction of IO from any primitive P that can be oracle- xed constructed from the . In particular, we will do so for P that is the extended IRHWE primitive. Our task is thus twofold: (1) to prove that P can be oracle- xed constructed from and (2) to show a simulatable compilation procedure that compiles out from any IO construction. The rst task is proven in Section 7.5.3 and the second task is proven in Section 7.5.4. By Lemma 7.2.7, this would imply the separation result of IO from P and prove Theorem 7.5.3.

Our oracle, which is more formally de ned in Section 7.5.2, resembles an idealized version of a homomorphic witness encryption scheme, which makes the construction of extended IRHWE straightforward. As a result, the main challenge lies in showing a simulatable compilation procedure for IO that satis es De nition 7.2.2 in this idealized model, and therefore, it is instructive to look at how the compilation process works and what challenges are faced with dealing with oracle .

High-level Compiler Structure and Challenges

Here we brie y discuss the general structure of the proposed compiler before going over the issues that arise when dealing with how to compile out . In this ideal model, we will aim to compile out both the decryption and evaluation procedures to reduce the oracle into a basic random oracle. Note, however, that while we will be using an instance-revealing oracle (and therefore we can run V on the revealed attribute to discover its indirect queries similar to the IRWE case), we still have to handle those hidden queries issued by the evaluation function F.

The Challenge Faced with Homomorphic Witness Encryption. Similarly to the case of compiling out rIHWE, since is an extended oracle, the algorithm F is allowed to

issues queries to any subroutine in . As a result, during the learning process of the compilation procedure, a query to Eval_F might be asked for which queries issued by F would remain hidden (see De nition 7.4.12) and therefore may a ect correctness when a new execution of $\widehat{B}(x)$ hits one of those queries.

Resolving the Challenge. We resolve this problem by canonicalizing the emulation and learning processes as well as the obfuscation B so that we force the obfuscation B, upon issuing a query q of the form $\text{Eval}_F(c_1, ..., c_k)$, to explicitly reveal any queries asked by F by having it call F $(m_1, ..., m_k)$ if it knows *all* of the encryptions $\text{Enc}(a_i, m_i) = c_i$. However, we still need to consider the case that, when executing $\widehat{B}$ (x), at least one of the encryptions are unknown, in which case, we simply emulate the answer of q to be set of uniformly *random* ciphertexts (which we will categorize as fake ciphertexts). We argue that unless a Dec query is asked to decrypt one of the fake ciphertexts then the real execution's distribution is close to an ideal execution's distribution. We prove that due to the learning procedure, such a decryption query has a low probability of happening.

7.5.2 The Ideal Model

In this section, we de ne the distribution of our ideal randomized oracle.

Definition 7.5.4 (Random Instance-revealing Witness Homomorphic Encryption Oracle)**.** Let V be a universal circuit-evaluator that takes instance a and witness w and either accepts or rejects. Furthermore, let F be a universal circuit evaluator that takes as input a sequence of messages $m_1, ..., m_k$ to output some value m'. Let $H : \{0,1\} \to \{0,1\}^{n^2}$ be a public random hash function. We de ne the following *random instance-revealing witness homomorphic encryption* (rIRWHE) oracle $_{V;F;n} = (\text{Enc}, \text{Rev}, \text{Dec}_V, \text{Eval}_F)$ as follows:

> Enc: $\{0,1\}^n \to \{0,1\}^{2n}$ is a random injective function. We will use the Enc oracle to encrypt a message $m \in \{0,1\}^{n^2}$ with respect to attribute $a \in \{0,1\}^{n^2}$ to obtain a ciphertext $c \in \{0,1\}^{2n}$.
>
> For any $c \in \{0,1\}^{2n}$, we call c valid if there exists x such that $\text{Enc}(x) = c$ and fake otherwise.
>
> Rev: $\{0,1\}^{2n} \to \{0,1\}^{n^2} \cup \{?\}$ is a function that, given an input $c \in \{0,1\}^{2n}$, would output the corresponding a for which $\text{Enc}((a, m)) = c$. If there is no such preimage then it outputs $?$ instead.
>
> Dec_V: $\{0,1\}^s \to \{0,1\}^n \cup \{?\}$: Given $w \in \{0,1\}^k, c \in \{0,1\}^{2n}$ as inputs where $k = \text{poly}(n)$ and $s = k + 2n$, $\text{Dec}(w, c)$ allows us to decrypt the ciphertext c and get $x = (a, m)$ as long as the predicate test is satis ed on (w, a). More formally, do as follow:
>
> 1. If @ x such that $\text{Enc}(x) = c$, output $?$. Otherwise, continue to the next step.

2. Find x such that Enc(x) = c.
3. If V (w, a) = 0 output $?$. Otherwise, output x = (a, m).

Eval$_F$: $f0,1g^{2nk}$ $?$ $f0,1g^{2np}$: Given a sequence of inputs $c_1, ..., c_k$, this subroutine performs the following:

1. If for some i 2 $[k]$ @ x_i = (a_i, m_i) for Enc(x_i) = c_i, then output $?$. Otherwise, continue to the next step.
2. For each i 2 k, nd x_i = (a_i, m_i) such that Enc(x_i) = c_i (this is an ine cient process). Let $fa'_1, ..., a'_pg$ be the set of distinct attributes embedded in $x_1, ..., x_k$ where p k.
3. Run b = F $(m_1, ..., m_k)$
4. Output $fc'_ig_{i=1}^p$ where, for all i 2 $[p]$, c'_i = Enc(x'_i), x'_i = (a'_i, b_ijjh_i), where the value h is de ned as h_i = $H(i, a_1, ..., a_k, m_1, ..., m_k)$ and b_i is the i'th share of b.

We de ne a query-answer pair resulting from query q to subroutine T 2 fEnc, Rev, Dec$_V$, Eval$_F$$g$ with some answer β as $(q$ $?$ $\beta)_T$. For simplicity, when n is clear from the context, we use = (Enc, Rev, Dec$_V$, Eval$_F$) to refer to $_{V;F;n}$ for a xed n.

Remark 7.5.5. Note that for the sake of achieving the most general case of extended primitives from this oracle, we will allow both F and V to have any oracle gates to = (Enc, Dec$_V$, Rev, Eval$_F$).

7.5.3 Homomorphic Witness Encryption exists relative to

In this section, we show how to construct a semantically-secure extended extractable IRHWE with corresponding universal-circuit evaluators V and F relative to = (Enc, Rev, Dec$_V$, Eval$_F$). More speci cally, we will show how to construct a primitive (in the oracle model) that is simpler to prove the existence of and yet still implies EIRHWE.

Definition 7.5.6 (Extended Extractable One-way Homomorphic Witness Encryption (ex-EO-HWE)). Let V be a universal circuit-evaluator that takes instance a and witness w and either accepts or rejects. Furthermore, let F be a universal circuit evaluator that takes as input a sequence of messages $m_1, ..., m_k$ to output some value m'. For any given security parameter κ, an *extended extractable one-way homomorphic witness encryption* scheme consists of the following PPT algorithms P = (Enc, Rev, Dec$_V$, Eval$_F$) de ned as follows:

Enc($a, m, 1$) : given an instance a 2 $f0,1g$, message m 2 $f0,1g$, and security parameter κ (and randomness as needed) it outputs c 2 $f0,1g$.

Rev(c) : given ciphertext c returns the underlying attribute a 2 $f0,1g$.

Dec$_V$(w, c) : given ciphertext c and \witness" string w, it outputs a message m' 2 $f0,1g$.

$\text{Eval}_F(c_1, ..., c_k)$: given ciphertexts $c_1, ..., c_k$, outputs another sequence of ciphertexts $(c'_1, ..., c'_p)$ where $p \quad k$.

An extended extractable one-way homomorphic witness encryption scheme satis es the same completeness properties of De nition 7.5.2 but with the extractability property replaced with the following:

Extractable One-Wayness: For any PPT adversary A and polynomial $p_1(.)$, there exists a PPT (black-box) straight-line extractor E and a polynomial function $p_2(.)$ such that the following holds. For any security parameter κ, k = poly(κ), and for all a, if:

$$\Pr\left[A(1^\kappa, c) = m \mid m \xleftarrow{\$} \{0,1\}^k, c \quad \text{Enc}(a, m, 1^\kappa)\right] \quad \frac{1}{p_1(\kappa)}$$

Then:

$$\Pr[E^A(a) = w \wedge V^P(w, a) = 1] \quad \frac{1}{p_2(\kappa)}$$

Construction 7.5.7 (Extended Extractable One-way Homomorphic Witness Encryption)**.** Let V be a universal circuit-evaluator that takes instance a and witness w and either accepts or rejects. Furthermore, let F be a universal circuit evaluator that takes as input a sequence of messages $m_1, ..., m_k$ to output some value m'. For any security parameter κ and oracle sampled according to De nition 7.5.4, we will implement an extended EOHWE scheme P for messages $m \in \{0,1\}^k$ where k = poly(κ) as follows:

WEnc$(a, m, 1^\kappa)$: Given $a \in \{0,1\}^*$, message $m \in \{0,1\}^k$ and security parameter 1^κ, output Enc(x) where $x = (a, m)$.

WDec(w, c) : Given witness w and ciphertext c, let $x' = \text{Dec}_V(w, c)$. If $x' \neq \perp$, parse as $x' = (a', m')$ and output m'. Otherwise, output $\perp$.

WEval$(c_1, ..., c_k)$: Given ciphertexts $c_1, ..., c_k$, it outputs $\text{Eval}_F(c_1, ..., c_k)$.

Remark 7.5.8 (From one-wayness to indistinguishability.)**.** We note that the primitive ex-EOHWE, which has one-way security, can be used to build an indistinguishability-based ex-IRHWE. For any a, since Enc$(a, .)$ is a random injective function (and hence one-way) we have that, by the Goldreich-Levin thoerem [GL89], there exists a hardcore predicate $b = \langle r, r' \rangle$ for the one-way function $\text{Enc}'(a, r, r') := (\text{Enc}(a, r), r')$. Now, to encrypt a one-bit message b under some attribute a, we would output the ciphertext $c = (\text{Enc}(a, r), r')$ where $r, r' \quad \{0,1\}^k$ are randomly sampled conditioned on $b = \langle r, r' \rangle$. Furthermore, to perform an evaluation $\text{Eval}'_{F'}(c_1, ..., c_p)$ over a set of ciphertexts of the form $c_i = (c'_i, r'_i)$ where $c'_i = \text{Enc}(a, r_i)$, we will rst choose r'_F uniformly at random then run $\text{Eval}_F(c'_i, ..., c'_p)$ where F is de ned such that it outputs random r_F for which $\langle r_F, r'_F \rangle = F'(\langle r_1, r'_1 \rangle, ..., \langle r_p, r'_p \rangle)$. We then output $c_F = (\text{Enc}(a, r_F), r'_F)$ as the new evaluated ciphertext.

Lemma 7.5.9. *Construction 7.5.7 is a correct and subexponentially-secure oracle- xed implementation (De nition 3.1.4) of an extended extractable one-way homomorphic witness encryption in the ideal oracle model.*

To prove the security of this construction, we will show that if there exists an adversary A against scheme P (in the oracle model) that can invert an encryption of a random message with non-negligible advantage then there exists a (xed) deterministic straight-line extractor E with access to = (Enc, Rev, Dec_V, Eval_F) that can nd the witness for the underlying instance of the challenge ciphertext.

Suppose A is an adversary in the inversion game with success probability ϵ. Then the extractor E would works as follows: given a as input and acting as the challenger for adversary A, it chooses $m \xleftarrow{\$} \{0,1\}$ uniformly at random then runs A (1 , c) where c WEnc($a, m, 1$) is the challenge. Let Q_A be the set of query-answer pairs for the encryption queries that A will ask. Queries issued by A are handled by E as follows:

> To answer any query Enc(x) asked by A, it forwards the query to the oracle and returns some answer c. Add $(x \mapsto c)_{\text{Enc}}$ to Q_A.
>
> To answer any query Rev(c) asked by A, it forwards the query to the oracle and returns some answer a.
>
> To answer any query $\text{Dec}_V(w, c)$ asked by A, the extractor rst issues a query Rev(c) to get some answer a. If $a \neq \perp$, it would execute V (w, a), forwarding queries asked by V to similar to how it does for A and adds any resulting query-answer pairs to Q_A. Finally, it forwards the query $\text{Dec}_V(w, c)$ to to get some answer x. If $a = \perp$, it returns $\perp$ to A otherwise it returns x and adds $(x \mapsto c)_{\text{Enc}}$ to Q_A if $x \neq \perp$.
>
> To answer any query $\text{Eval}_F(c_1, ..., c_k)$ asked by A, the extractor rst issues queries Rev(c_1), ..., Rev(c_k) to get a sequence of answers $(a_1, ..., a_k)$. Let $(a'_1, ..., a'_p)$ be the set of unique attributes from $(a_1, ..., a_k)$. If $a'_i \neq \perp$ for all $i \in [p]$, it would check if there exists $((a_i, m_i) \mapsto c_i)_{\text{Enc}}$ for all $i \in [k]$ in Q_A. If so, it proceeds to run $m' \leftarrow F(m_1, ..., m_k)$, forwarding queries asked by F to , then nally returns $\{c'_i\}_{i=1}^{p}$ where $c'_i = \text{Enc}(a'_i, b_i \| h_i)$, b_i is the i'th random additive share of m' and the value of h_i is equal to $H(i, a_1, ..., a_k, m_1, ..., m_k)$. Finally, it adds all of $(x'_i \mapsto c'_i)_{\text{Enc}}$ to Q_A where $x'_i = (a'_i, b_i \| h_i)$.
>
> However, if $(x_i \mapsto c_i)_{\text{Enc}} \notin Q_A$ for some $i \in [k]$, and in particular if $c_i = c$ for some i, then the extractor would instead emulate the ciphertext evaluations c'_i by encrypting a random message for each di erent attribute a'_i (since it does not know the underlying plaintext and cannot run F) and we call c'_i fake ciphertexts. Note that by the collision resistance property of H, A would not be able to distinguish between such fake ciphertexts and real evaluated ones.

While handling the queries made by A, if a decryption query $\text{Dec}_V(w, c)$ is issued by A (perhaps even indirectly as a result of running V or F) for some c such that Rev(c) = Rev(c) (this includes either decrypting c directly or another c' having the same attribute as c that was perhaps generated as a result of an evaluation query with c as one of the inputs), the extractor will pass this query to , and if the result of the decryption is $x \neq \perp$ then the extractor will halt execution and output w as the witness for instance x = (a, m). Otherwise,

if after completing the execution of A, no such query was asked then the extractor outputs $?$. We prove the following lemma.

Lemma 7.5.10. *For any PPT adversary A and instance a, if there exists a non-negligible function $\epsilon(.)$ such that:*

$$\Pr\left[A\ (1^\kappa, c) = m \mid m \xleftarrow{\$} \{0,1\}^k, c \leftarrow \mathsf{WEnc}(a, m, 1^\kappa)\right] \geq \epsilon(\kappa) \tag{7.5}$$

Then there exists a PPT extractor E such that:

$$\Pr\left[E^{\mathcal{A}}(a) = w \wedge V(w, a) = 1\right] \geq \epsilon(\kappa) - \mathrm{negl}(\kappa) \tag{7.6}$$

Proof. Let A be an adversary satisfying Equation (7.5) above and let AdvWin be the event that A succeeds in the inversion game. Furthermore, let ExtWin be the event that the extractor succeeds in extracting a witness (as in Equation (7.6) above). Note that, by our construction of extractor E, when E executes A, it perfectly simulates the oracle for A assuming that several bad events do not happen. Let B_1 be the bad event that A asks $\mathsf{Dec}(w, c_1)$ or $\mathsf{Eval}(c_1, ..., c_k)$ where $(x_i, c_i)_{\mathsf{Enc}} \in Q_A$ for some $i \in [k]$. Let B_2 be the bad event that A asks $\mathsf{Dec}(w, c_1)$ or $\mathsf{Eval}(c_1, ..., c_k)$ where $(x_i, c_i)_{\mathsf{Enc}} \in Q_A$ for some i is a hidden query that was asked due to a previous query $\mathsf{Eval}_F(y_1, ..., y_k)$ and $y_j = c$ or is otherwise fake for some j (so F could not be executed by the extractor). As long as $B = (B_1 \vee B_2)$ does not happens then the extractor will perfectly simulate A's view.

Note that for the special case of FHE, as long as B does not happen then all fake ciphertexts have the same attribute as c since the only case that $\mathsf{Eval}(c_1, ..., c_k)$ would generate a fake ciphertext is when $\mathsf{Rev}(c_i) = \mathsf{Rev}(c)$ for all i. If A issues $\mathsf{Dec}_V(w, c_i^\emptyset)$ for some fake ciphertext $c_i^\emptyset$ then the extractor would return the answer returned by even though that answer might be a random incorrect answer. However, this is safe to do since once A successfully decrypts a fake ciphertext (which must have an attribute equal to the attribute of c), the extractor wins immediately.

For the more general case where a fake ciphertext $c_i^\emptyset$ generated by $(c_1^\emptyset, ..., c_k^\emptyset) \leftarrow \mathsf{Eval}(c_1, ..., c_k)$ queries may have different attributes (depending on the attribute value of the input ciphertexts), we argue that decrypting $c_i^\emptyset$ where $\mathsf{Rev}(c_i^\emptyset) \neq \mathsf{Rev}(c)$ will not affect the distribution of the view of A. Recall that, by the design of the extractor, $c_i^\emptyset$ would be an encryption of a random message when in the ideal world it would have been an encryption of the i'th random additive share of $F(m_1, ..., m_k)$. However, by the security of the random secret sharing, we can argue that, without knowing all the other secret shares, the distributions $\mathsf{Dec}(w, c_i^\emptyset)$ and U_{n-2} are statistically close.

As a result of the above, the event B reduces to finding an image of a sparse random injective function, which is negligibly hard to accomplish. Now we show that whenever A wins then E must win as well. Observe that:

$$\Pr_{;m}[\overline{\mathsf{ExtWin}}] \leq \Pr_{;m}[\overline{\mathsf{ExtWin}} \wedge \mathsf{AdvWin} \wedge \overline{B}] + \Pr_{;m}[\overline{\mathsf{AdvWin}}] + \Pr_{;m}[B]$$

Since $\Pr[\mathsf{AdvWin}] \geq \epsilon$ for some non-negligible function ϵ and $\Pr[B] \leq \mathrm{negl}(\kappa)$, it suffices to show that $\Pr[\overline{\mathsf{ExtWin}} \wedge \mathsf{AdvWin} \wedge \overline{B}]$ is negligible. Note that this event is equivalent to

saying that the adversary succeeds in the inversion game but never asks a query of the form $\mathsf{Dec}_{\mathsf{V}}(w, c')$ such that $\mathsf{Rev}(c') = \mathsf{Rev}(c)$ and for which the answer is $x \neq ?$ so the extractor fails to recover the witness. For simplicity of notation define $\mathsf{Win} := \overline{\mathsf{ExtWin}} \wedge \mathsf{AdvWin} \wedge \overline{B}$.

We will show that, with overwhelming probability over the choice of oracle , the probability of Win happening is negligible. That is, we will prove the following claim:

Claim 7.5.11. *For any negligible δ,* $\Pr\left[\Pr_m[\mathsf{Win}] \quad p_{\overline{\delta}}\right] \quad p_{\overline{\delta}}$

Proof. We list all possible queries that A could ask and argue that these queries do not help A in any way without also forcing the extractor to win as well. Specifically, we show that for any such A that satisfies the event Win, there exists another adversary $\widehat{A}$ that depends on A and also satisfies the same event but does not ask any decryption or evaluation queries (only encryption queries). This would then reduce to the standard case of inverting a random injective function, which is known to be hard over the randomness of the oracle. We define the adversary $\widehat{A}$ as follows. Upon executing A, it handles the queries issued by A as follows:

If A asks a query of the form $\mathsf{Enc}(x)$ then $\widehat{A}$ forwards the query to to get the answer.

If A asks a query of the form $\mathsf{Rev}(c)$ then, since B does not happen, it must be the case that $((a, m)\ ?\!?\ c)_{\mathsf{Enc}} \in Q_A$ and therefore $\widehat{A}$ returns a.

If A asks a query of the form $\mathsf{Dec}(w, c)$ or $\mathsf{Dec}(w, c')$ where c' has the same attribute as c then w must be a string for which $\mathsf{V}(w, a) = 0$ or otherwise the extractor wins, which contradicts that $\overline{\mathsf{ExtWin}}$ happens. If that is the case, since w is not a witness, $\widehat{A}$ would return $?$ to A after running $\mathsf{V}(w, a)$ and answering its queries appropriately.

If A asks a query of the form $\mathsf{Dec}(w, c')$ for some $c' \neq c$ then, since B does not happen, it must be the case that A has asked a (direct or indirect) visible encryption query $\mathsf{Enc}(x') = c'$. Therefore, $\widehat{A}$ would have observed this encryption query and can therefore run $\mathsf{V}(w, a')$ and return the appropriate answer (x or $?$) depending on the answer of V.

If A asks a query of the form $\mathsf{Eval}_{\mathsf{F}}(c_1, ..., c_k)$ and $((a_i, m_i)\ ?\!?\ c_i)_{\mathsf{Enc}} \in Q_A$ for all i then $\widehat{A}$ can compute $\beta = F(m_1, ..., m_k)$, handling its queries, returning the correct encryptions of each share of β, and adding the resulting new encryptions to Q_A.

If A asks a query of the form $\mathsf{Eval}_{\mathsf{F}}(c_1, ..., c_k)$ and $c_i = c$ or is otherwise a fake ciphertext then $\widehat{A}$ returns encryptions of random messages under the same attributes as the input ciphertexts (without running F). Note that, while an ideal execution would generate hidden encryption queries from running F, since B does not happen, these hidden encryption queries will not be decrypted with high probability.

Given that $\widehat{A}$ perfectly emulates A's view, the only possibility that A could win the inversion game is by asking $\mathsf{Enc}(x) = c$ and hitting the challenge ciphertext. By a standard averaging argument, we find that since $\Pr_{,m}[\mathsf{Win}] \quad \delta(\kappa)$ for some negligible δ then $\Pr[\Pr_m[\mathsf{Win}] \quad p_{\overline{\delta}}] \quad 1 \quad p_{\overline{\delta}}$, which yields the result.

□

To conclude the proof of Lemma 7.5.10, we can see that the probability that the extractor wins is given by Pr[ExtWin] 1 Pr[$\overline{\text{ExtWin}}$ ^ AdvWin ^ $\overline{B}$] Pr[$\overline{\text{AdvWin}}$] Pr[B] $\epsilon(\kappa)$ negl(κ) where ϵ is the advantage to A.

□

Proof of Lemma 7.5.9. It is clear that the Construction 7.5.7 is correct. Furthermore, by Lemma 7.5.10, it also satis es the extractability property. □

7.5.4 Compiling out from IO

In this section, we show a simulatable compiler for compiling out . We formalize the approach outlined in Section 7.5.1 while making use of Lemma 7.2.4, which allows us to compile out in two phases: we rst compile out part of to get an approximately-correct obfuscator $\widehat{\text{IO}}^R$ in the random oracle model, and then use the result of [CKP15] to compile out R and get an obfuscator $\widetilde{\text{IO}}$ in the plain-model. Since we are applying this lemma only a constant number of times (in fact, just twice), security should still be preserved. Speci cally, we will prove the following claim:

Lemma 7.5.12. *Let R @ be a random oracle where "$\vee$" denotes a sub-model relationship (see De nition 7.2.1). Then the following holds:*

> *For any IO in the ideal model, there exists a simulatable compiler with correctness error $\epsilon < 1/200$ for it that outputs a new obfuscator in the random oracle R model.*
>
> *For any IO in the random oracle model, there exists a simulatable compiler with correctness error $\epsilon < 1/200$ for it that outputs a new obfuscator in the plain model.*

Proof. The second part of Lemma 7.5.12 follows directly from [CKP15], and thus we focus on proving the rst part of the claim. Before we start describing the compilation process, we present the following de nition of canonical executions that is a property of algorithms in this ideal model and dependent on the oracle being removed.

Definition 7.5.13 (Canonical executions)**.** We de ne an oracle algorithm A relative to rIRHWE to be in canonical form if it satis es all of the following requirements:

> Before asking any $\text{Dec}_{\text{V}}(w, c)$ query for a valid c query, A would rst get a Rev(c) then run V (w, a) on its own, making sure to answer any queries of V using .
>
> After asking a query $\text{Dec}_{\text{V}}(w, c)$ for which the returned answer is some message m ~~6=~~ ?, issue a query a Rev(c) followed by Enc(x) where x = (a, m).
>
> Before A asks a query $\text{Eval}_{\text{F}}(c_1, ..., c_k)$ for which it knows the corresponding encryptions c_1 = Enc$(a_1, m_1), ..., c_k$ = Enc(a_k, m_k), it would rst call b F $(m_1, ..., m_k)$, making sure to answer any queries of F using . Let $(a'_1, ..., a'_p)$ be the distinct attributes from $(a_1, ..., a_k)$ where p k and b_i be the i'th share of b where i 2 $[p]$. A would then issue calls to Enc(x'_i) where for all i 2 $[p], x'_i = (a_i, b_i \| H(i, a_1, ..., a_k, m_1, ..., m_k))$.

After A asks a query $\mathsf{Eval}_{\mathsf{F}}(c_1, ..., c_k)$ for which the returned answer is a sequence of ciphertexts $c'_1, ..., c'_p$ where $p \geq k$, issue queries $\mathsf{Rev}(c'_i)$ for all $i \in [p]$.

Note that any oracle algorithm A can be easily modified into a canonical form by increasing its query complexity by at most a polynomial factor (since V and F are PPT algorithms).

We now proceed to present the construction of the random oracle model obfuscator $\mathsf{IO} = (\mathsf{iO}, \mathsf{Ev})$ that, given an obfuscator in the model, would compile out and emulate queries to Rev, Dec and Eval while forwarding any Enc queries to R. Throughout this process, we assume that the obfuscator iO and evaluation procedure Ev are all canonicalized according to Definition 7.5.13. For these canonical algorithms, let $l_O, l_B = \mathrm{poly}(\kappa)$, respectively, be the number of queries asked by the ideal-model obfuscator iO and the evaluation Ev to the rIRHWE oracle .

The new obfuscator $\widehat{\mathsf{IO}}^R$ in the random oracle model

Let $R = \{R_n\}_{n \in \mathbb{N}}$ be the (injective) random oracle where $R_n : \{0,1\}^n \to \{0,1\}^{2n}$. Given a δ-approximate obfuscator $\mathsf{IO} = (\mathsf{iO}, \mathsf{Ev})$ in the rIRWHE oracle model, we construct an $(\delta + \epsilon)$-approximate obfuscator $\widehat{\mathsf{IO}} = (\widehat{\mathsf{iO}}, \widehat{\mathsf{Ev}})$ in the random oracle model.

Subroutine $\widehat{\mathsf{iO}}^R(C)$:

1. *Emulation phase*: Given input $C \in \mathcal{C}_n$, emulate $\mathsf{iO}(C)$ and let B be the output of this emulation. Let T_O be the transcript of this phase and initialize $Q_O := Q(T_O) = ?$. For every query q asked by $\mathsf{iO}(C)$:

   Call $(\rho_q, W_H) \leftarrow \mathtt{EmulateCall}(Q_O, q)$ as shown in Algorithm 3 and return the answer in ρ_q

   Add ρ_q and W_H to Q_O

   Note that we will not pass any query-answer pairs Q_O asked by iO to the output of this new obfuscation. All of the query-answer pairs in Q_O will only be used for emulation purposes in the following step.

2. *Learning phase*: Let Q_B be the set of all visible queries that will be learned (which will be passed on to the obfuscated circuit) and let Q_B^h be the set of hidden queries that we will keep track of for emulation purposes (which will not be made public to the obfuscated circuit). Set $m = 4\ell_O/\epsilon$ where $\ell_O \leftarrow |\mathsf{iO}|$ represents the number of queries asked by iO. Choose $t \xleftarrow{\$} [m]$ uniformly at random then for $i = 1, ..., t$ do the following:

   Choose $z_i \xleftarrow{\$} \{0,1\}^{|C|}$ uniformly at random

   Run $\mathsf{Ev}(B, z_i)$. For every query q asked by $\mathsf{Ev}(B, z_i)$:

   - Call $(\rho_q, W_H) \leftarrow \mathtt{EmulateCall}^R(Q_O \cup Q_B \cup Q_B^h, q)$ and return the answer in ρ_q. Add ρ_q to Q_B and add W_H to Q_B^h.

Algorithm 3: `EmulateCall`

Input: Query-answer set Q, query q
Oracle: Random Oracle R, Hash function H
Output: (ρ_q, W_H) where ρ_q is a query-answer pair containing the answer of query q and W_H is the set of indirect query-answer pairs (those added by F)
Begin:
Initialize W_H = ?
if q is a query of type Enc(x) **then**
  Set ρ_q = $(x \; ?! \; R(x))_{\text{Enc}}$
end
if q is a query of the form Rev(c) **then**
  if $\exists (x \; ?! \; c)_{\text{Enc}} \in Q$ where $x = (a, m)$ **then**
    Set ρ_q = (c, a)
  else
    Set ρ_q = $(c, ?)$
  end
end
if q is a query of the form $\text{Dec}_V(w, c)$ **then**
  if $\exists (x \; ?! \; c)_{\text{Enc}} \in Q$ for some $x = (a, m)$ **then**
    Emulate $b \quad V \; (w, a)$ while handling its queries using `EmulateCall`
    if $b = 1$ **then**
      Set ρ_q = $((w, c) \; ?! \; x)_{\text{Dec}}$
    else
      Set ρ_q = $((w, c) \; ?! \; ?)_{\text{Dec}}$
    end
  else
    Set ρ_q = $((w, c) \; ?! \; ?)_{\text{Dec}}$
  end
end
[Continued next page...]

Algorithm 3: EmulateCall (Continued)

if q is a query of the form $\mathsf{Eval}_{\mathsf{F}}(c_1, ..., c_k)$ **then**
  For all $i \in [k]$, get (ρ_{a_i}, W_a) $\leftarrow$ $\mathsf{EmulateCall}(Q, \mathsf{Rev}(c_i))$ then add ρ_{a_i} to Q
  Let $\{a'_1, ..., a'_p\}$ be the distinct attributes of $\{a_1, ..., a_k\}$
  If $a_i = ?$ for some i then return $((q, ?, ?)_{\mathsf{Eval}}, W_H)$
  if $\forall i \exists ((a_i, m_i), c_i)_{\mathsf{Enc}} \in Q$ and c_i are valid **then**
    Emulate $b \leftarrow \mathsf{F}(m_1, ..., m_k)$
    for each query q_{F} asked by F **do**
      $(\rho_{\mathsf{F}}, W_{\mathsf{F}}) \leftarrow \mathsf{EmulateCall}^R(Q \cup W_H, q_{\mathsf{F}})$
      $W_H = W_H \cup (\rho_{\mathsf{F}} \cup W_{\mathsf{F}})$
    end
    For all $a'_i \in \{a'_1, ..., a'_p\}$, set $c'_i \leftarrow \mathsf{Enc}(a'_i, b_i \| H(i, a_1, ..., a_k, m_1, ..., m_k))$ and add to W_H
  else
    For all $a'_i \in \{a'_1, ..., a'_p\}$, set $c'_i \leftarrow \mathsf{Enc}(a'_i, r_i)$ where $r_i \xleftarrow{\$} \{0,1\}^{n/2}$ and add to W_H
  end
  Set $\rho_q = ((c_1, ..., c_k), (c'_1, ..., c'_p))_{\mathsf{Eval}}$
end
Return (ρ_q, W_H)

Since Ev is a canonical algorithm, in addition to the direct queries asked by $\mathsf{Ev}(B, z_i)$ to , the resulting query set Q_B would also include the indirect queries that have resulted from $\mathsf{Ev}(B, z_i)$ running $\mathsf{V}(w, a)$. Furthermore, if a query $\mathsf{Eval}_{\mathsf{F}}(c_1, ..., c_k)$ is issued where $\mathsf{Enc}((a_i, m_i)) = c_i$ then the indirect queries asked by $\mathsf{F}(m_1, ..., m_k)$ will be made visible to the final execution so long as the query-answer pairs $((a_i, m_i), ?, c_i)_{\mathsf{Enc}}$ for all i were directly asked beforehand (i.e. were in Q_B).

3. The output of the random oracle model obfuscation algorithm $\widehat{\mathsf{iO}}^R(C)$ will be $\widehat{B} = (B, Q_B)$ where Q_B is the set of visible (direct/indirect) query-answer pairs.

Subroutine $\widehat{\mathsf{Ev}}^R(\widehat{B}, z)$**:** Initialize $Q_{\widehat{B}} = ?$ to be the set of queries asked when evaluating $\widehat{B}$. To evaluate $\widehat{B} = (B, Q_B)$ on a new random input z, we emulate $\mathsf{Ev}(B, z)$. For every query q asked by $\mathsf{Ev}(B, z)$:

Run and set $(\rho_q, W_H) = \mathsf{EmulateCall}^R(Q_B \cup Q_{\widehat{B}}, q)$ then return the answer in ρ_q.

Add ρ_q and W_H to $Q_{\widehat{B}}$

Remark 7.5.14 (Fake ciphertexts)**.** Throughout the circuit evaluation $\widehat{\mathsf{Ev}}(\widehat{B}, z)$ subroutine, note that if a query of the form $\mathsf{Eval}_{\mathsf{F}}(c_1, ..., c_k)$ is asked where at least one of the input ciphertexts c_i is not known to the evaluator, the answer to the query will be simulated as a

sequence of encryptions of random messages (since at least one the underlying messages is unknown and therefore F cannot be computed). From here on, we refer to such simulated ciphertexts as *fake ciphertexts*.

The running time of $\widehat{\text{iO}}$. We note that the running time of the new obfuscator $\widehat{\text{iO}}$ remains polynomial time since we are emulating the original obfuscation once followed by a polynomial number m of learning iterations. Furthermore, while we are indeed working with an oracle where the universal circuit evaluators V and F can have oracle gates to subroutines of , we emphasize that in our framework of extended primitives, the e ective running times of V and F, which we are executing during EmulateCall, remain a strict polynomial in the size of the original circuits and so the issue of exponential or in nite recursive calls is non-existent.

Proving Approximate Correctness. Let $Q_S = (Q_O \,[\, Q_B \,[\, Q_B^h \,[\, Q_{\widehat{B}})$ be the set of all query-answer pairs asked during the emulation, learning, and nal execution phases. We de ne the following hybrid experiments to show that the correctness of the new random-oracle model obfuscator that simulates an execution of a -model obfuscation is su ciently close to the correctness of an ideal execution of the -model obfuscation. We denote the distribution of Hybrid i as $H^i = (Q_O^i, Q_B^i, Q_B^{i,h}, Q_{\widehat{B}}^i)$.

Hybrid 0: This is the real experiment, which is represented as the construction in Section 7.5.4.

Hybrid 1: This is the same as Hybrid 0 except for two di erences:

- The queries asked during emulation (Step 1) and learning (Step 2) answered relative to a true oracle instead of emulating them using EmulateCall.
- We modify the nal execution $\widehat{\text{Ev}}(\widehat{B}, z)$ such that, when we execute Ev (B, z), we would forward all queries to instead of using EmulateCall *except* when the query q is of the form Eval$(c_1, ..., c_k)$ where $(x_i \stackrel{?}{=} c_i)_{\text{Enc}} \stackrel{?}{\notin} Q_B \,[\, Q_{\widehat{B}}$ for some c_i, in which case we *do not* forward this query to and use the simulated answer returned by EmulateCall$(Q_B \,[\, Q_{\widehat{B}}, q)$ as the answer instead (which does not run F and returns fake ciphertexts instead).

Hybrid 2: This is the ideal experiment, where *all* the queries asked during emulation, learning and nal execution are answered relative to a true oracle . That is, all queries whose answers are emulated using EmulateCall throughout the obfuscation and nal evaluation phases will instead be answered directly using .

Claim 7.5.15. $(H^0, H^1) \quad \frac{\epsilon}{2} + \text{negl}(\kappa)$

Proof. We rst observe that the emulation and learning processes of Hybrid 0 is a statistical simulation of the true oracle. The only event that can cause a discrepancy between these two hybrids during the emulation and learning phases is when a query of the form Rev(c),

$\mathrm{Dec}_V(w, c)$, or Eval$(c_1, ..., c_k)$ is issued but the emulator has never issued an encryption query that maps to an input of those queries (hitting a valid ciphertext without knowing the underlying plaintext). In that case, in Hybrid 0, the answer to such a query will be $\perp$ as de ned by `EmulateCall` whereas the answer in Hybrid 1 will be consistent with the true oracle and return the correct answer. However, the probability of that happening in Hybrid 1 (over the randomness of oracle) is at most $(2^n - p)/(2^{2n} - p) < 1/2^{n-1}$ = negl(κ) given that $p = tl_B l_O$ = poly(κ).

Next, we need to show that the distribution of query-answer pairs in the nal execution phase is close. In Hybrid 0, the execution of $\widehat{B}$ emulates $\mathrm{Ev}(B, z)$ on a random input z and emulates the answers of all queries using `EmulateCall`, whereas in Hybrid 1 we answer all queries of $\mathrm{Ev}(B, z)$ using the true except for Eval$(c_1, ..., c_k)$ queries for which $(x_i, c_i)_{\mathsf{Enc}} \notin Q_B \cup Q_{\widehat{B}}$ for some c_i. In essence, in Hybrid 0, we can think of the execution as $\widehat{\mathrm{Ev}}(B, z)$ where $\widehat{\ }$ is the oracle simulated using just $Q_B \cup Q_{\widehat{B}}$, and in Hybrid 1, we can think of the execution as $\widetilde{\mathrm{Ev}}(B, z)$ where $\widetilde{\ }$ is the oracle simulated using just $Q_B \cup Q_{\widehat{B}}$ and except for evaluation queries of the aforementioned type (which generate fake ciphertexts). We will identify the events that di erentiate between the executions $\widehat{\mathrm{Ev}}(B, z)$ and $\widetilde{\mathrm{Ev}}(B, z)$.

Assume that the query-answer pairs so far during the execution of $\widehat{B}$ are the same in both hybrids. That is, we start with some $Q^0_{\widehat{B}} = Q^1_{\widehat{B}} = Q_{\widehat{B}}$ for the sake of proving inductively that any subsequent query-answer pairs are closely distributed between the two hybrids. Let q be a new (possibly indirect) query that is being asked by $\mathrm{Ev}(B, z)$. We present a case-by-case analysis of all possible query types to identify the cases that can cause discrepancies between the two hybrids:

1. If q is a query of type Enc(x), then it will be answered the same in both hybrids using the random oracle R in Hybrid 0 and using the true oracle in Hybrid 1.

2. If q is of type Rev(c) for which there exists $((a, m), c)_{\mathsf{Enc}} \in Q_B \cup Q_{\widehat{B}}$ or $(c, a)_{\mathsf{Rev}} \in Q_B \cup Q_{\widehat{B}}$ then the answer to q would be a in both hybrids.

3. If q is of type Dec(w, c) for which there exists $(x, c)_{\mathsf{Enc}} \in Q_B \cup Q_{\widehat{B}}$ then the answer to q would be determined the same (x is returned) in both hybrids.

4. If q is of type Eval$(c_1, ..., c_k)$ for which there exists $((a_i, m_i), c_i)_{\mathsf{Enc}} \in Q_B \cup Q_{\widehat{B}}$ then the answer to q would be determined the same in both hybrids. In particular, both hybrids will run $F(m_1, ..., m_k)$, adding any indirect queries to $Q_{\widehat{B}}$ and answering q with the a set of correctly generated ciphertexts.

5. If q is of type Rev(c_1), Dec(w, c_1), or Eval$(c_1, ..., c_k)$ such that there exists $(x_i, c_i)_{\mathsf{Enc}} \notin Q_S$ for some $i \in [k]$ then this means that we are attempting to decrypt a ciphertext for which we have not encrypted before (either directly or indirectly). In Hybrid 1, since Enc is a sparse random injective function, the answer would be $\perp$ with overwhelming probability as c would be invalid, and in Hybrid 0 the answer will also be $\perp$ by the de nition of $\widehat{\mathrm{Ev}}(\widehat{B}, z)$ in this hybrid. Note that in both hybrids, there will

not be any indirect queries in $Q_{\widehat{B}}$ as a result of this query since the ciphertexts that are being queried on are deemed invalid and neither V or F will be executed.

6. If q is of type Eval($c_1, ..., c_k$) such that there exists $(x_i \; ? \; c_i)_{\mathsf{Enc}} \; 2 \; Q_S \; n \; (Q_B \; [\; Q_{\widehat{B}})$ for some $i \; 2 \; [k]$ then this means that we are attempting to decrypt a ciphertext that was not learned. However in both hybrids we will simulate the answer as encryptions of random messages and in both hybrids there will not be any hidden indirect queries in $Q_{\widehat{B}}$ as a result of this query.

7. Suppose q is of type Rev(c) that is not determined by $Q_B \; [\; Q_{\widehat{B}}$ in Hybrid 0 and yet is determined by Q_S in Hybrid 1. We list the di erent cases here that may cause problems:

   (a) **Bad Event 1:** The query-answer pair $((a, m) \; ? \; c)_{\mathsf{Enc}}$ is in $Q_O \; n \; (Q_B \; [\; Q_{\widehat{B}})$. That is, we are for the rst time revealing a valid ciphertext that was encrypted in Step 1 because we failed to learn the answer of this query during the learning phase of Step 2. In that case, in Hybrid 0, the answer would be *?* since we do not know the corresponding attribute a whereas in Hybrid 1 it would use to produce the answer consistent with $Q_O \quad Q_S$ and output a.

   (b) **Bad Event 2:** The query-answer pair $((a, m) \; ? \; c)_{\mathsf{Enc}}$ is in $Q_B^h \; n \; (Q_B \; [\; Q_{\widehat{B}})$. That is, we are for the rst time revealing a ciphertext from a hidden encryption query in Step 2. In that case, in Hybrid 0, the answer would be *?* since we do not know the corresponding attribute a whereas in Hybrid 1 it would use the answer consistent with $Q_B^h \quad Q_S$ and output a.

8. Suppose q is of type Dec(w, c) that is not determined by $Q_B \; [\; Q_{\widehat{B}}$ in Hybrid 0 and yet is determined by Q_S in Hybrid 1. We list the di erent cases here that may cause problems:

   (a) If q is of type Dec(w, c) such that $((a, m) \; ? \; c)_{\mathsf{Enc}} \; 2 \; Q_S \; n \; (Q_B \; [\; Q_{\widehat{B}})$ and V (w, a) = 0 then this means that we are attempting to decrypt a ciphertext for which the decryption will fail. The answer will be *?* in Hybrid 0 by the de nition of `EmulateCall` and is also *?* in Hybrid 1 since we are using the real oracle . Note that in both hybrids, while there will be indirect queries (as a result of running V (w, a)), such queries will be visible in both worlds.

   (b) **Bad Event 3:** The query-answer pair $((a, m) \; ? \; c)_{\mathsf{Enc}}$ is in $Q_O \; n \; (Q_B \; [\; Q_{\widehat{B}})$ and V (w, a) = 1. That is, we are for the rst time decrypting a valid ciphertext that was encrypted in Step 1 because we failed to learn the answer of this query during the learning phase of Step 2. In that case, in Hybrid 0, the answer would be *?* since we do not know the corresponding message $x = (a, m)$ whereas in Hybrid 1 it would use to produce the answer consistent with $Q_O \quad Q_S$ and output x.

   (c) **Bad Event 4:** The query-answer pair $((a, m) \; ? \; c)_{\mathsf{Enc}}$ is in $Q_B^h \; n \; (Q_B \; [\; Q_{\widehat{B}})$ and V (w, a) = 1. That is, we are decrypting a ciphertext from a hidden encryption

query in Step 2. In that case, in Hybrid 0, the answer would be $?$ since we do not know the corresponding message $x = (a, m)$ whereas in Hybrid 1 it would use the answer consistent with $Q_B^h \quad Q_S$ and output x.

(d) **Bad Event 5:** The query-answer pair $\rho_e = ((c_1, ..., c_k) \; ? \; (c_1', ..., c_p'))_{\mathsf{Eval}}$ is in $Q_S \cap Q_{\widehat{B}}$, $c = c_i'$ for some $i \in [p]$, $\exists (x_j \; ? \; c_j)_{\mathsf{Enc}} \notin Q_B \cup Q_{\widehat{B}}$ and $\mathsf{V}(w, a_i') = 1$ where $a_i' = \mathsf{Rev}(c_i')$. That is, we are decrypting one of the (non-fake) ciphertexts created from an evaluation query for which the underlying plaintexts of at least one of the ciphertext inputs to the evaluation are unknown to the evaluator. In that case, in Hybrid 0, the answer would be $?$ since we do not know the corresponding message underlying c whereas in Hybrid 1 it would use the answer supplied by and output the correct answer. We argue that that this bad event can be reduced to the previous bad events.

If $\rho_e \in Q_O$ then this implies that $(x \; ? \; c_i')_{\mathsf{Enc}} \in Q_O$ due to our canonicalization procedure. As a result, this means that we are for the first time successfully decrypting a ciphertext from Step 1, which reduces to Case 8b.

If $\rho_e \in Q_B \cup Q_B^h$ then this implies that $(x \; ? \; c_i')_{\mathsf{Enc}} \in Q_B^h$ due to our emulation procedure. As a result, this means that we are for the first time successfully decrypting a ciphertext from a hidden encryption query in Step 2, which reduces to Case 8c.

(e) The query-answer pair $((c_1, ..., c_k) \; ? \; (c_1', ..., c_p'))_{\mathsf{Eval}}$ is in $Q_{\widehat{B}}$, $c = c_i'$ for some $i \in [p]$, $\exists (x_j \; ? \; c_j)_{\mathsf{Enc}} \notin Q_B \cup Q_{\widehat{B}}$ and $\mathsf{V}(w, a_i') = 1$ where $a_i' = \mathsf{Rev}(c_i')$. That is, we are decrypting a fake ciphertext created from an evaluation query for which the underlying plaintexts of at least one of the ciphertext inputs to the evaluation are unknown to the evaluator. However, since the query was generated during the evaluation phase, the fake ciphertext c was also generated during the evaluation and therefore $(x \; ? \; c)_{\mathsf{Enc}}$ is known where $x = (a \| r)$ for some random r. Thus, in both Hybrids, the answer would be x.

For input x, let $E_1(x)$ be the event that Case 8b or Case 7a happens, and $E_2(x)$ be the event that Case 8c or Case 7b happens. Assuming that event $E(x) = (E_1(x) \_ E_2(x))$ does not happen, both experiments will proceed identically the same and the output distributions of $\widehat{\mathsf{Ev}}(B, z)$ and $\widetilde{\mathsf{Ev}}(B, z)$ will be statistically close.

Claim 7.5.16. $\Pr_x[E_1(x)] \quad \epsilon/4$.

Proof. We bound this event in Hybrid 1. Recall that E_1 is the event that during the final execution of the obfuscation $\widehat{B}$ on a random input, a successful decryption or reveal query is asked for the first time for a ciphertext that was generated during the obfuscation phase but was not learned during the learning phase. For all $i \in [t]$, let $Q_{B_i}' = (Q_{B_i} \cup Q_{B_i}^h) \setminus Q_O$ be the set of query-answer pairs generated by the i'th evaluation $\mathsf{Ev}(B, z_i)$ during the learning phase (Step 2) and are also generated during the obfuscation emulation phase (Step 1). Note that, since $t > \ell_O$ and $Q_{B_i}' \quad Q_{B_{i+1}}'$, the number of learning iterations that increase the size

of the set of learned obfuscation encryption queries is at most ℓ_O. We say $t \xleftarrow{\$} [m]$ is bad if it is the case that $Q^0_{B_t} \subsetneq Q^0_{B_{t+1}}$ (i.e. t is an index of a learning iteration that increases the size of the learned obfuscation encryption queries). This would imply that after t learning iterations in Hybrid 1, the real execution $Q^0_{\widehat{B}} := Q^0_{B_{t+1}}$ would contain a new unlearned query that was in Q_O. Thus, given that $m = 4\ell_O/\epsilon$, the probability (over the selection of t) that t is bad is at most $\ell_O/m < \epsilon/4$. □

Claim 7.5.17. $\Pr_x[E_2(x) \mid \overline{E}_1(x)] \leq \epsilon/4 + \mathrm{negl}(\kappa)$

Proof. Recall that E_2 is the event that during the final execution of the obfuscation $\widehat{B}$ on a random input, a decryption or reveal query is asked for a ciphertext that was generated during the learning phase but was part of the hidden set and therefore never publicized. We will in fact first bound this event (assuming E_1 does not happen) in Hybrid 2.

For all $i \in [t]$, let $Q_{B_i} = (Q_{B_i} \setminus Q^h_{B_i})$ be the set of (public and hidden) query-answer pairs generated by the i'th evaluation $\mathsf{Ev}(B, z_i)$ during the learning phase (Step 2) and $Q_{\widehat{B}}$ be the query-answer pairs of the final evaluation of the obfuscation. Note that in Hybrid 2, the real oracle is used and therefore each execution of $\mathsf{Ev}(B, z_i)$ is independent of the other executions. In particular, we can think of the final evaluation as iteration $t + 1$ of the learning phase with query-answer pairs $Q_{B_{t+1}} = Q_{\widehat{B}}$. Now, we can think of an intermediate hybrid with distribution $H^{2'}$ where the executions are randomly permuted in a way such that execution $t + 1$ in Hybrid 2 is now the first execution in Hybrid $2'$. Note that, since the executions are independent, the distributions H^2 and $H^{2'}$ are equivalent and it remains to show that E_2 is unlikely to happen in Hybrid $2'$. Specifically, we need to show that the first execution of the learning phase (with query-answer pairs $Q_{B_{t+1}}$) does not decrypt a ciphertext generated by any previous hidden queries in the learning phase. However, since this is the first execution, there exists no hidden queries from previous executions and it suffices to argue that it does not decrypt a ciphertext generated by any hidden queries $Q^h_{B_{t+1}}$ from its own execution.

Since E_1 does not happen and since the probability that $\mathsf{Ev}(B, z_{t+1})$ decrypts a ciphertext that was never encrypted during the whole process is negligible, the only ciphertexts that this execution will attempt to decrypt are ones that were generated during this very same execution. Therefore in Hybrid 2, E_2 only happens with negligible probability when a ciphertext that was never encrypted before is being decrypted. Now using Claim 7.5.18 we can conclude that this event in Hybrid 1 happens with probability at most $\epsilon/4 + \mathrm{negl}(\kappa)$.

□

Thus, using Claims 7.5.16 and 7.5.17, we find that $\mathrm{Adv}(H^0, H^1) \leq \epsilon/2 + \mathrm{negl}(\kappa)$.

□

Claim 7.5.18. $\mathrm{Adv}(H^1, H^2) \leq \frac{\epsilon}{4} + \mathrm{negl}(\kappa)$

Proof. The only difference between the two hybrids is that in Hybrid 1, we do not forward to any queries of the form $\mathsf{Eval}(c_1, \ldots, c_k)$ for which we do not know all of c_i, and instead we will use some simulated (fake) ciphertexts. Whereas in Hybrid 2, we do indeed execute

them using the real oracle. Thus, while there may be hidden queries as a result of running F in Hybrid 2, the counterparts in Hybrid 1 do not have such hidden queries and are instead replaced with fake ciphertexts. As a result, we have to consider the cases where Rev, Dec or Eval might use any hidden ciphertexts in Hybrid 2 and/or fake ciphertexts in Hybrid 1.

Assume that the query-answer pairs so far during the execution of $\widehat{B}$ are the same in both hybrids. That is, we start with some $Q^1_{\widehat{B}} = Q^2_{\widehat{B}} = Q_{\widehat{B}}$ for the sake of proving inductively that any subsequent query-answer pairs are closely distributed between the two hybrids. Let q be a new (possibly indirect) query that is being asked by Ev(B, z).

1. Determined queries: If q is a query of type Enc(x) then it will answered the same in both hybrid using R. If q is of type Rev(c) for any c then the answer will be the same in both hybrids. If q is of type Dec(w, c) for any non-fake ciphertext c where $(x \leftarrow c)_{\mathsf{Enc}} \in Q_S \cap Q^h_{\widehat{B}}$, then the answer will be the same in both hybrids. If q is of type Eval($c_1, ..., c_k$) whose answer can be determined by $Q_B \cup Q_{\widehat{B}}$ then the answer will be the same in both hybrids.

2. **Bad Event 1:** If q is of type Eval($c_1, ..., c_k$) such that there exists $(x_i \leftarrow c_i)_{\mathsf{Enc}} \in Q_S \cap (Q_B \cup Q_{\widehat{B}})$ for some $i \in [k]$ then this means that we are evaluating on some ciphertext that was not learned. In this case in Hybrid 1 the answer would be a set of fake ciphertexts generated by encrypting random messages under their respective attributes. However, in Hybrid 2 the answer would be a set of ciphertexts generated by encrypting shares of the true evaluation. Note that by the collision resistance property of H and the injectivity of Enc, the two sets of ciphertexts (the fake and non-fake) will be statistically close.

3. Suppose that q is of type Dec(w, c). We list the different cases that can cause problems:

   (a) **Bad Event 2:** There exists a query-answer pair $(x \leftarrow c)_{\mathsf{Enc}} \in Q^h_{\widehat{B}} \cap (Q_B \cup Q_{\widehat{B}})$ that was generated as one of the hidden queries issued by some evaluation query during execution $\widehat{\mathsf{Ev}}(\widehat{B}, z)$. In that case, in Hybrid 1, the answer would be $\perp$ with overwhelming probability since the encryption was never created there, whereas in Hybrid 2, the answer would be x. However we note that the probability of that happening in Hybrid 1 is negligible since it is equal to the event that a ciphertext was hit that has not been generated before.

   (b) **Bad Event 3:** The query-answer pair $((c_1, ..., c_k) \leftarrow (c'_1, ..., c'_p))_{\mathsf{Eval}}$ is in $Q_{\widehat{B}}$, $c = c'_i$ for some $i \in [p]$, $\exists((a_j, m_j) \leftarrow c_j)_{\mathsf{Enc}} \notin Q_B \cup Q_{\widehat{B}}$ and $V(w, a'_i) = 1$ where $a'_i = \mathsf{Rev}(c'_i)$. That is, we are decrypting one of the (fake) ciphertexts from the output of an evaluation query for which the underlying plaintexts of at least one of the ciphertext inputs to the evaluation query is unknown. Note that each fake ciphertext c'_i corresponds to an encryption query $(x'_i \leftarrow c'_i)_{\mathsf{Enc}} \in Q_{\widehat{B}}$ for some $x'_i = (a'_i, r_i)$ and independently chosen random string r_i. In that case, in Hybrid 1, the answer would be x'_i whereas in Hybrid 2 it would use the answer supplied by $\perp$ and output the answer of the evaluation $x_i = (a'_i, m'_i)$ where m'_i is an evaluated answer based on the messages encrypted by $(c_1, ..., c_k)$.

For the case of FHE (or multi-key FHE), decrypting any fake ciphertext implies decrypting for the first time all of the input ciphertexts, which must have originated from Q_O or Q_B^h since we need to know witnesses for all the input ciphertexts $(c_1, ..., c_k)$ in order to decrypt c. Thus, this event is similar to Case 8d of Claim 7.5.15 and so we refer to the proof there for this case.

For the more general case of spooky encryption, we may decrypt a fake ciphertext without needing to know witnesses for all the input ciphertexts and so we cannot argue that we are decrypting for the first time some input ciphertext. Nevertheless, all we obtain by decrypting a single (or even p 1) ciphertexts are individual shares of the evaluation result, which are identically distributed to random values r_i as long as we do not have witnesses for *all* the input ciphertexts (if we do then we are back to the multi-key FHE case).

Thus, using the same reasoning as Claims 7.5.16 and 7.5.17, we find that (H^1, H^2) $\epsilon/4$ + negl(κ).

□

Using Claims 7.5.15 and 7.5.18 we can finally conclude the proof of approximate correctness since we can now show that (H^0, H^1) + (H^1, H^2) $3\epsilon/4$ + negl(κ) ϵ.

Proving Security. To show that the resulting obfuscator is secure, it suffices to show that the compilation process represented as the new obfuscator's construction is simulatable. We show a simulator S (with access to) that works as follows: given an obfuscated circuit B in the ideal model, it runs the learning procedure as shown in Step 2 of the new obfuscator $\widehat{\text{iO}}$ to learn the heavy queries Q_B then outputs $\widehat{B}$ = (B, Q_B). Note that this distribution is statistically close to the output of the real execution of $\widehat{\text{iO}}$ and, therefore, security follows.

□

7.6 Primitives Implied by Our Variants of Witness Encryption

In this section, we will show that extended witness encryption (or one of its variants) implies extended versions of several encryption primitives of the all-or-nothing flavor. Recall that we previously argued that witness encryption and its considered variants do not imply indistinguishability obfuscation in a monolithic way. This allows us to conclude a monolithic separation between the studied all-or-nothing encryption primitives and indistinguishability obfuscation.

7.6.1 Extended Predicate Encryption

In this section, we will show a monolithic construction of extended PE from EIHWE and one-way function. Since, one-way functions imply digital signatures in a black-box manner,

we will assume that there exists a signature scheme (Gen, Sign, Verify) where these procedures only make black-box calls to the OWF. Let (WEnc, WDec_V) be an EIHWE scheme where V is the universal circuit evaluator (see De nition 2.1.1) that is allowed to have OWF gates, and WEnc and WDec_V gates. The relation V is de ned such that V$((k, sk_k), C_{\text{MPK};a;m})$ = $C_{\text{MPK};a;m}(k, sk_k)$ where circuit $C_{\text{MPK};a;m}$ is de ned as follows:

$$C_{\text{MPK};a;m}(k, sk_k) := (\text{P}(k, a) = 1 \wedge \text{Verify}(\text{MPK}, k, sk_k) = 1)$$

Then, our extended PE scheme for the universal class of predicates $P_{K;A}$ (computed by Turing Machine P as in De nition 2.6.4) and message space M corresponds to the PPT algorithms (Setup, KGen, Enc, Dec_P) de ned below. Note that we are constructing an extended PE scheme and therefore P is allowed to have OWF, Setup, KGen, Enc, and Dec_P gates.

Setup(1): outputs (MPK, MSK) where (MPK, MSK) Gen(1).

KGen(MSK, k): given k 2 K and the master secret key MSK 2 $f0, 1g^n$, outputs the decryption key sk_k = Sign(MSK, k). If $k = \epsilon$, it outputs ϵ.

Enc(MPK, (m, a)): outputs ciphertext c = WEnc($C_{\text{MPK};a;m}, (a, m)$) where $C_{\text{MPK};a;m}$ is de ned above.

$\text{Dec}_P(sk_k, c)$: given a secret key sk_k for k 2 K and a ciphertext c, obtain $(C_{\text{MPK};a;m}, a, m)$ = $\text{WDec}_V((k, sk_k), c)$, and output m.

Specifying V given P. Since we are constructing extended predicate encryption, our P is allowed to have gates of OWFs, Setup, KGen, Enc, and Dec_P planted in it. Next, observe from the scheme that V contains one P gate and one Verify gate. Also note that P does not make any calls to V.

Next we argue that both the P gate and the one Verify gate can be simpli ed to OWF, WEnc, and WDec_V gates that V is allowed to have. We modify V as follows.

1. We embed P in V. Note that V now has all the gates P had and a Verify gate. Speci cally, V has OWF, Verify, Setup, KGen, Enc, and Dec_P gates.

2. Next, we syntactically replace Setup gates with Gen gates, KGen gates with Sign gates, Enc gates with WEnc gates and Dec_P gates with WDec_V gates. Note that this replacement will require some additional code changes in V which all depend on the above described construction.

3. Next, since we use a known construction of the signature scheme (Gen, Sign, Verify) and it is black-box in the use of OWFs, we can replace these gates by just OWF gates along with some additional code that depends on the used signature scheme. Observe that we have reduced all gates in V to just OWF, WDec, and WDec_V gates.

Lemma 7.6.1. *Extended fully-secure PE scheme (De nition 2.6.4) is implied by extended EIHWE and OWFs.*

Proof. Correctness of the above scheme follows from the observation that a witness encryption ciphertext along with a signature on k, namely sk_k, always yields the encrypted message whenever P(k, a) = 1. Next we prove security.

Let A be a computationally bounded adversary that asks a polynomial number of secret-key queries and breaks the security of the PE scheme. In other words, for some polynomial $p(.)$,

$$\Pr[\mathrm{IND}_A^{\mathrm{PE}}(1^\kappa) = 1] \quad \frac{1}{2} + \frac{1}{p(\kappa)}$$

where $\mathrm{IND}_A^{\mathrm{PE}}$ is the following experiment (recalled from Figure 2.1):

Experiment $\mathrm{IND}_A^{\mathrm{PE}}(1^\kappa)$:
1. (MSK, MPK) Setup(1^κ))
2. (x_0, x_1) $A^{\mathrm{KGen}(\mathrm{MSK},\cdot)}$(MPK) where $x_b = (a_b, m_b)$ for $b \in \{0,1\}$, $|a_0| = |a_1|$ and $|m_0| = |m_1|$ and for each prior query k we require that P(k, a_0) = P(k, a_1) = 0
3. $b \xleftarrow{\$} \{0,1\}$
4. c Enc(MPK, x_b)
5. b' $A^{\mathrm{KGen}(\mathrm{MSK},\cdot)}$(MPK, c) where for each query k we require that P(k, a_0) = P(k, a_1) = 0
6. Output 1 if $b = b'$ and 0 otherwise.

At a high level, we would like to use the attacker A above to contradict the security of the signature scheme. Towards this, our approach will be to use the extractor E of ex-EIHWE that translates A's ability to distinguish ciphertexts to extracting a witness, which in our case is a forgery. However, a technical issue is that E might rewind A and A might fail on those correlated executions. We solve this issue by a standard probability argument as explained next.

Let T be the transcript of the inputs to A in steps 1 and 2 above | namely, T = (MPK, $\{sk_k\}, x_0, x_1$) where $\{sk_k\}$ is the collection of secret-keys that A obtains in step 2 above. Then, for the machine $A(T)$ we de ne the following experiment:

Experiment $\mathrm{IND}_{A(T)}^{\text{PE-partial}}(1^\kappa, \mathrm{MPK}, \mathrm{MSK})$:
1. $b \xleftarrow{\$} \{0,1\}$
2. c Enc(MPK, x_b)
3. b' $A(T)^{\mathrm{KGen}(\mathrm{MSK},\cdot)}$(MPK, c) where for each query k we require that P(k, a_0) = P(k, a_1) = 0
4. Output 1 if $b = b'$ and 0 otherwise.

By an averaging argument we have that if $\Pr[\mathrm{IND}_A^{\mathrm{PE}}(1^\kappa) = 1] \quad \frac{1}{2} + \frac{1}{p(\kappa)}$ then for a non-negligible fraction of choices of (T, MPK, MSK) we have that $\Pr[\mathrm{IND}_{A(T)}^{\text{PE-partial}}(1^\kappa, \mathrm{MPK}, \mathrm{MSK}) =$

1] $\frac{1}{2} + \frac{1}{2p(\kappa)}$.

Towards contradiction, we will now show that given A we can break the existential unforgeability of the signature scheme under chosen message attack. On receiving the public-key MPK from the challenger we prepare the machine $A(T)$ (by including signatures on keys of A's choice in T). Next by De nition 7.4.2 we have that there exists an a black-box extractor $E^{A(T)}(a_0, a_1)$ which with non-negligible probability outputs a signature on k for some choice of k such that P(k , a_0) = 1 or P(k , a_1) = 1. This breaks the security of the signature scheme.

Note that during the execution of $E^{A(T)}(a_0, a_1)$, A could ask for more signature queries for any k such that P(k, a_0) = P(k, a_1) = 0. However, these queries can be answered with the help of the challenger.

□

7.6.2 Extended Spooky Encryption

In this section, we will show a monolithic construction of extended spooky encryption from IRHWE and PRG. Let (WEnc, WDec_V, $\mathsf{WEval}_\mathsf{F}$) be an IRHWE scheme where F is a universal circuit evaluator and G be a length doubling pseudorandom generator. The relation V is de ned such that V(w, C_{MPK}) = $C_{\mathsf{MPK}}(w)$ where circuit C_{MPK} is de ned as follows:

$$C_{\mathsf{MPK}}(w) := (\mathsf{MPK} = G(w))$$

Then our extended spooky encryption scheme (Setup, Enc, $\mathsf{Eval}_{\mathsf{F}_s}$, Dec) (where F_s denotes the universal circuit evaluation function for the spooky encryption scheme) is de ned as follows:

Setup(1): outputs (MPK = $G(s)$, MSK = s) where s is a randomly sampled seed of the pseudorandom generator.

Enc(MPK, m): outputs ciphertext c = WEnc(C_{MPK}, m) where C_{MPK} is de ned above.

$\mathsf{Eval}_{\mathsf{F}_s}(f, (\mathsf{MPK}_1, c_1), \ldots, (\mathsf{MPK}_t, c_t))$: output $\mathsf{WEval}_\mathsf{F}(f, c_1, \ldots c_t)$.

Dec(MSK, c): given a secret key MSK and a ciphertext c, output WDec_V(MSK, c).

Specifying V. Observe that V in the extended spooky encryption scheme above only has one PRG gate that the extended IRHWE scheme supports.

Specifying F_s. Since we are constructing extended spooky encryption, F_s is allowed to have gates of Setup, Enc, Dec, and $\mathsf{Eval}_{\mathsf{F}_s}$ planted in it. We start by observing that for any F_s we can syntactically replace each Setup gate with a PRG gate, each Enc gate with a WEnc gate, each $\mathsf{Eval}_{\mathsf{F}_s}$ gate with a $\mathsf{WEval}_\mathsf{F}$ gate, and each Dec gate with a WDec_V gate. This change requires some additional code depending on the above described construction. This leaves us with only PRG, WEnc, WDec_V, and $\mathsf{WEval}_\mathsf{F}$ gates that F of extended IRWHE supports. Thus the F_s supports gates as required by extended spooky encryption scheme.

Lemma 7.6.2. *Extended spooky encryption scheme (De nition 2.6.8) is implied by extended IRHWE and PRG.*

Proof. Correctness of the above scheme follows directly from the correctness of the IRHWE scheme. For security, we need to show that for any PPT adversary A, there exists a negligible function negl(.) such that the following holds:

$$|\Pr[A(\text{MPK}, \text{Enc}(\text{MPK}, 0)) = 1] \quad \Pr[A(\text{MPK}, \text{Enc}(\text{MPK}, 1)) = 1]| \quad \text{negl}(\kappa)$$

where (MSK, MPK) Setup(1) and the probability is over the randomness of A and Enc. This follows by a simple argument as follows.

H_0: This hybrid corresponds to the distribution (MPK, Enc(MPK, 0)) where (MSK, MPK) Setup(1).

H_1: In this hybrid, we change how MPK is generated. Instead of generating MPK as the output of G, we sample MPK as a uniformly random string.

Indistinguishability between H_0 and H_1 follows from the pseudorandomness property of the PRG G.

H_2: In this hybrid, instead of encrypting 0 we encrypt 1. Namely, the ciphertext is generated as Enc(MPK, 1) instead of Enc(MPK, 0).

Since MPK is a chosen uniformly at random therefore except with negligible probability we have that $\forall w, \text{MPK} \notin G(w)$. Hence, by the security of witness encryption we have that the distribution Enc(MPK, 1) is computationally indistinguishable from Enc(MPK, 0).

H_3: In this hybrid, we switch back to generating MPK honestly (i.e., as the output of the pseudorandom generator).

Indistinguishability between H_2 and H_3 follows from the pseudorandomness property of the PRG G.

Observe that hybrid H_3 corresponds to the distribution (MPK, Enc(MPK, 1)) where (MSK, MPK) Setup(1). This concludes the proof. □

7.6.3 Extended Attribute Based FHE

In this section, we will show a construction of extended Attribute Based FHE (ABFHE) from extended extractable instance-revealing homomorphic witness encryption (EIRHWE) and one-way functions (OWFs). Since one-way functions imply digital signatures in a black-box manner, we will assume that there exists a signature scheme (Gen, Sign, Verify) where these procedures only make black-box calls to the OWF. Let (WEnc, WRev, WDec_V, WEval_F) be the ex-EIRHWE primitive and V and F are universal circuit evaluators that are allowed to

have OWF gates, and WEnc, WDec_V, and WEval_F oracle gates. The relation V is de ned such that $V((k, sk_k), C_{MPK;a}) = C_{MPK;a}(k, sk_k)$ where circuit $C_{MPK;a}$ is de ned as follows:

$$C_{MPK;a}(k, sk_k) := (P(k, a) = 1 \wedge \mathsf{Verify}(MPK, k, sk_k) = 1)$$

Then, our extended ABFHE scheme for the universal predicate class $P_{K;A}$ (computed by machine P) and message space M corresponds to the PPT algorithms (Setup, KGen, Enc, Dec_P, Eval_{F_s}) de ned below. Note that we are constructing an extended ABFHE, therefore P and F_s are allowed to have OWF, Setup, KGen, Enc, Dec_P and Eval_{F_s} gates.

Setup(1): outputs (MPK, MSK) where (MPK, MSK) Gen(1).

KGen(MSK, k): given k 2 K and the master secret key MSK 2 $f0,1g^n$, outputs the decryption key sk_k = Sign(MSK, k). If $k = \epsilon$, it outputs ϵ.

Enc(MPK, (m, a)): outputs ciphertext c = WEnc($C_{MPK;a}, m$) where $C_{MPK;a}$ is de ned above.

$\mathsf{Eval}_{F_s}(f, MPK, c_1, \ldots, c_t)$: If WRev($c_1$) = WRev($c_2$) = = WRev($c_t$)[6] then output $\mathsf{WEval}_F(f, c_1, \ldots, c_t)$ and otherwise output ?.

$\mathsf{Dec}_P(sk_k, c)$: given a secret key sk_k for k 2 K and a ciphertext c, output $\mathsf{WDec}_V((k, sk_k), c)$.

Specifying V given P. Since we are constructing extended ABFHE, our P is allowed to have gates of OWF, Setup, KGen, Enc, Eval_{F_s} and Dec_P planted in it. Observe from the scheme that V contains one P gate and one Verify gate. Next we argue that both these gates can be simpli ed to OWF, WEnc, WDec_V, and WEval_F gates that ex-EIRHWE supports. We modify V as follows.

1. We embed P in V. Note that V now has all the gates P had and a Verify gate. Speci cally, V has OWF, Verify, Setup, KGen, Enc, Eval_{F_s} and Dec_P gates.

2. Next, we syntactically replace Setup gates with Gen gates, KGen gates with Sign gates, Enc gates with WDec_V gates, Eval_{F_s} gates with WEval_F gates and Dec_P gates with WDec_V gates. Note that this replacement will require some additional code changes in V which all depend on the above described construction.

3. Next, since we use a known construction of the signature scheme (Gen, Sign, Verify) and it is black-box in the use of OWFs, we can replace these gates by just OWF gates along with some additional code that depends on the used signature scheme. Observe that we have reduced all gates in V to just OWF, WEnc, WEval_F and WDec_V gates.

[6]Here, WRev is the reveal function of the IRWHE that we ignored in the rest of the exposition. This function is only needed at this point.

Specifying F given F_s. Since we are constructing extended ABFHE, Eval_{F_s} is allowed to have gates of OWFs, $\mathsf{Setup}, \mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec}_P$, and Eval_{F_s} planted in it. We start by observing that for any F_s we can syntactically replace each Setup and KGen gate with a OWF gates, each $\mathsf{Enc}, \mathsf{Eval}_{F_s}$ and Dec_P with WEnc, WEval_F and WDec_V, respectively.In making this change we need to add some additional code to F depending on the above described construction and the code of the signature scheme. This leaves us with only OWF, WEnc, WDec_V, and WEval_F gates that F of ex-EIRHWE supports. Thus the above described construction supports gates as required by the extended ABFHE scheme.

Lemma 7.6.3. *Extended fully-secure ABFHE scheme is implied by ex-EIRHWE and OWFs.*

Proof. Note that security game of the ABFHE scheme does not involve the Eval_{F_s} function, which only a ects functionality. Therefore, the proof of security of this claim is essentially the same as the proof of Lemma 7.6.1. The only di erence is that we are now using ex-EIRHWE instead of ex-EIHWE. □

Chapter 8

Monolithic Separation of IO from Functional Encryption

8.1 Introduction

After proving that several \all-or-nothing" encryption primitives are insu cient for giving us IO, we now turn to investigate the possibility of basing IO on an even more expressive primitive: functional encryption (FE). What is currently known about the power and complexity of functional encryption is the following:

1. *Compact single-key FE is known to imply IO.* Recent results by Ananth and Jain [AJ15] and Bitansky and Vaikuntanathan [BV15] show how to base IO on a compact FE scheme | namely, a (single-key) FE scheme (see De nition 2.6.9) for which the running time of the encryption circuit is independent of the size of the supported function class. Furthermore, the construction works even if the ciphertext is weakly compact, i.e. the length of the ciphertext grows sub-linearly in the circuit size but is allowed to grow arbitrarily with the depth of the circuit.
2. *Positive results on single-key FE.* The construction of IO from compact single-key FE puts us in close proximity to primitives known from standard assumptions. One prominent work, is the (single-key) functional encryption scheme of Goldwasser, Kalai, Popa, Vaikuntanathan and Zeldovich [GKP+13] that is based on LWE. Interestingly, this encryption scheme is *weakly compact* for Boolean circuits. However, in this scheme the ciphertext grows additionally with the output length of the circuit for which the functional secret-key is given out. Hence, it is not known to imply IO.

In summary, the gap between the known single-key FE constructions from LWE and the single-key FE schemes known to imply IO (for the same ciphertext length properties) is only in the output length of circuit for which the functional secret-key is issued. In light of this, signi cant research continues to be invested towards realizing IO starting with various kinds of FE schemes (e.g. [BNPW16, BLP17]). This brings us to the question of what kind of functional encryption schemes realize (or cannot realize) IO.

8.2 Our Results

The main result of this chapter is to show that single-key FE schemes that support only functions with `short output' are incapable of producing IO even when non-black-box use of the FE scheme is allowed in certain ways. We speci cally use the *monolithic* framework developed in Chapter 6 which is equivalent to the fully black-box framework of [IR89,RTV04] applied to *extended primitives* (that can include all of their subroutines as gates inside circuits given to them as input). As we recall, this monolithic model captures the most commonly used non-black-box techniques in cryptography, including the ones used by Ananth and Jain [AJ15] and Bitansky and Vaikunthanathan [BV15] for realizing IO from FE. More formally, we prove the following theorem.

Theorem 8.2.1 (Main Result{Informal). *Assuming one-way functions exist and* $\mathbf{NP} \not\subseteq \mathbf{coAM}$*, there is no construction of IO from \short" output single-key FE where one is* allowed *to plant FE gates arbitrarily inside the circuits that are given to FE as input. An FE scheme is said to be \short" output if*

$$t(n, \kappa) \quad p(n, \kappa) \quad \omega(n + \kappa),$$

where n *is the plaintext length,* κ *is the security parameter,* p *is the ciphertext length (for messages of length* n*) and* t *is the output length of the functions evaluated on messages of length* n*.*

As a special case, the above result implies that single-key FE for boolean circuits and other single-key FE schemes known from standard assumptions are insu cient for IO in a monolithic way.

"Long-output" FE implies IO. Complementing this negative result, we show that above condition on ciphertext length t is almost tight. In particular, we show that a \long output" single-key FE | namely, a single-key FE scheme with $t = p + 1$ (supporting an appropriate class of circuits) is su cient for realizing IO. This construction is non-black-box (or, monolithic to be precise) and is obtained as a simple extension of the previous results of Ananth and Jain [AJ15] and Bitansky and Vaikuntanathan [BV15].

Fully Black-Box Separation of IO from FE. Finally, we show that some form of non-black-box techniques (beyond the fully black-box framework of [RTV04]) is necessary for getting IO from FE, *regardless* of the output lengths. Namely, we prove a fully black-box separation from FE to IO. Previously, Lin [Lin16] (Corollary 1 there) showed that the existence of such fully black-box construction from FE to IO would imply a construction of IO from LWE and constant-degree PRGs. Our result shows that no such fully black-box construction exists (but the possibility of IO from LWE and constant-degree PRGs remains open).

In close relation to this result, we make mention of the work of Ananth and Sahai [AS17] which, when combined with the result of Chapter 5, can be used to derive a special case of our

fully black-box separation of IO from FE. In more detail, [AS17] show that, assuming LWE, one can construct a sublinearly compact public-key FE scheme that makes black-box use of an underlying constant-degree encoding scheme (GES). On the other hand, in Section 5.4 we proved a fully black-box separation result of IO from $O(1)$-degree GES. The two results can be combined to imply a fully-black-box separation of IO from sublinearly compact FE. However, our result in this chapter is more general in the sense that we separate IO from even *fully compact* FE schemes, and we do so without the need for the additional LWE assumption.

On the Relation to [GKP+13]. Note that in Chapter 7 rules out the existence of monolithic IO constructions from attribute-based encryption (ABE) and the existence of monolithic IO constructions from fully homomorphic encryption (FHE). Furthermore, this result can be further broadened to separate IO from ABE *and* FHE in a monolithic way. One can then ask why the result in this chapter does not follow as a corollary from Chapter 7 and [GKP+13], where they construct single-key (non-compact) FE for general circuits from ABE and FHE.

We note that our result does not follow from the above observation for two reasons. First, the single-key FE construction of [GKP+13] also uses a garbling scheme in order to garble circuits with FHE decryption gates, whereas the impossibility in Chapter 7 does not capture such garbling mechanisms in the monolithic model. However, if one could improve the result of Chapter 7 in the monolithic model by adding a garbling subroutine that can accept ABE and FHE gates, then we can compose the results in Chapter 7 and [GKP+13] and obtain an impossibility of IO from t-bit output (non-compact) FE. Secondly, we note that this resulting t-bit output FE scheme has the property that $t \quad p/\text{poly}(\kappa)$ (i.e. the ciphertext size is a (polynomial) multiplicative factor of the output length of the function), whereas in this work we show the stronger impossibility of basing IO on single-key FE for output-length $t \quad p \quad \omega(\kappa)$.

8.3 Technical Overview

In order to demonstrate the ideas behind our impossibility, we start by recalling how the constructions of IO from FE [AJ15, BV15] work at a high level. Here, we present an `over-simpli ed' version of their construction which aims at describing the ideas presented here at a high level. In their construction, an obfuscation for a circuit $C : f0,1g \quad ! \quad f0,1g$ is a sequence of $\kappa+1$ functional keys of $\kappa+1$ instances of a single-key FE scheme along with a ciphertext c . These functional secret keys $\text{FSK}_1, \ldots, \text{FSK}_{\kappa+1}$ corresponding to public keys $\text{PK}_1 \ldots \text{PK}_{\kappa+1}$ are generated independently and the ciphertext c is obtained by encrypting the \empty" string under the rst public-key PK_1. The ciphertext does contain some cryptographic keys. This includes the secret-key of a secret-key encryption scheme denoted by s. The expression \empty" just refers to the fact that this ciphertext will be used to evaluate the obfuscation on all inputs. This will become clear in the following. The rst κ function keys implement the \bit-extension" functionality. That is, the i^{th} function key corresponds to a function

that takes in an $(i \quad 1)$-bit string $y \ 2 \ f0,1g^{i \ 1}$ as input and outputs encryptions of $(yj0)$ and $(yj1)$ under the public-key PK_{i+1}. Finally, the functional key FSK_{+1} corresponds to the circuit C.

To evaluate the obfuscated circuit on an input $x \ 2 \ f0,1g$, one does the following: decrypt c under FSK_1 to obtain encryptions of 0 and 1 denoted as c_0 and c_1 respectively. Next, it decrypts c_{x_1} (where x_1 is the rst bit of x) using FSK_2 and so on. Proceeding in this manner, in κ steps the evaluator obtains an encryption of x under PK_{+1} which can then be used to compute $C(x)$ using FSK_{+1}. Note that since the FE scheme does not hide the circuits embedded inside the functional secret-keys, therefore an encrypted version of the circuit C (under the key s also placed inside c) is embedded inside this key. The key s is passed along in each ciphertext. One can think of the construction as having a binary tree structure where evaluating the circuit on an input x corresponds to traversing along the path labeled x. Observe that, for this construction, each functional secret-key is generating an output that is larger than the size of the ciphertext that is decrypted using it. In particular, the decryption yields two ciphertexts | an output that is double the size of the input. On the other hand, in case the output of a functional secret key is \su ciently smaller" than a ciphertext, then this explosion in number of ciphertexts does not seem possible anymore. This is also the key to our impossibility. Roughly speaking, at the core of the proof of our impossibility result is to show that in this \small" output setting, the total number of ciphertexts that an evaluator can compute remains polynomially bounded. Turning this high level intuition into an impossibility proof requires several new ideas that we now elaborate upon below.

8.3.1 The Details of the Proof of Separation

As mentioned before, monolithic constructions of IO from FE are the same as fully black-box constructions of IO from *extended* FE which is a primitive that is similar to FE but it allows FE gates to be used in the circuits for which keys are issued. Therefore, to prove the separation, we can still use oracle separation techniques from the literature on black-box constructions [IR89].

In fact, for any candidate construction $IO^{( )}$ of indistinguishability obfuscation from extended FE, we construct an oracle O relative to which secure extended FE exists but the construction IO^O becomes insecure (against polynomial-query attackers). In order to do this, we will employ an intermediate primitive: a variant of functional witness encryption de ned by Boyle, Chung, and Pass [BCP14]. We call this variant customized FWE (cFWE for short) and show that (1) relative to our oracle cFWE exists, (2) cFWE implies extended FE in a black-box way, and that (3) the construction IO^O is insecure. We opted to work with this intermediate primitive of cFWE since it is conceptually easier to work with than an ideal FE oracle and allows us to leverage the techniques of Section 7.2.1 to prove our separation in a modular way. Now in order to get (1) we directly de ne our oracle O to be an idealized version of cFWE. To get (2) we use the power of cFWE.[1] To get (3) we rely on

[1] In fact, as shown in [BCP14], without our customization, the original FWE implies, not just IO itself, but even di-IO.

the fact that cFWE is *weakened* in a careful way so that it does not imply IO. Below, we describe more details about our idealized oracle for cFWE and how to break the security of a given candidate IO construction relative to this oracle. We first recap the general framework for proving separations for IO.

General recipe for proving separations for IO. Let I be our idealized cFWE oracle. We will use the general approach developed in Section 7.2.1 for breaking IO^I using a polynomial number of queries to the oracle (i.e. the step (3) above). Recall that the process involves \compiling out" the oracle I from the obfuscation scheme to get a new secure obfuscator $\mathsf{IO}^0 = (\mathsf{iO}^0, \mathsf{Ev}^0)$ in the *plain-model* that is only *approximately-correct*. Namely, by obfuscating $\mathsf{iO}^0(C) = B$ and running B over a *random* input we get the correct answer with probability $99/100$. Then, by the result of [BBF16], doing so implies a polynomial query attacker against IO^I in model I.

Note that this compiling out process (of I from IO^I) is not independent of the oracle being removed since different oracles may require different approaches to be emulated. However, the general high-level of the compiler that is used in Chapters 5 and 7, and we use here as well, is the same: The new plain-model obfuscator iO^0, given a circuit C to obfuscate would work in two steps. The first step of iO^0 is to emulate $\mathsf{iO}^I(C)$ (by simulating the oracle I) to get an ideal-model obfuscation B, making sure to `lazily' evaluate (emulate) any queries issued to I. The second step of the compiler is to learn the queries that are likely to be asked by $\mathsf{Ev}^I(B, x)$ for a uniformly random input x, denote by Q_B, which can be found by by emulating $\mathsf{Ev}^I(B, x_i)$ enough number of times for different uniformly random x_i. Finally, the output of iO^0 is the plain-model obfuscation $B^0 = (B, Q_B)$, where B is the ideal-model obfuscation and Q_B is the set of learned queries. To evaluate the obfuscation over a new random input x, we simply execute $\mathsf{Ev}^0(B, x) = \mathsf{Ev}^I(B, x)$ while emulating any queries to I consistently relative to Q_B.

Any compiler (for removing I from IO) that uses the approach describe above is in fact secure, because we only send emulated queries to the evaluator that could be simulated in the ideal world I. The challenge, however, is to prove the correctness of the new obfuscator. So we shall prove that, by having enough iterations of the learning process (in the learning step of iO^0), the probability that we ask an unlearned emulation query occurs with sufficiently small probability.

The Challenge Faced for Compiling Out Our Customized Functional Witness Encryption Oracle. When I is defined to be our idealized cFWE oracle, in order to prove the approximate correctness of the plain-model obfuscator, we face two problems.

1. **The Fuzzy Nature of FWE:** Unlike `all-or-nothing' primitives such as witness encryption and predicate encryption, functional witness encryption mechanisms allow for more relaxed decryption functionalities. In particular, decrypting a ciphertext does not necessarily reveal the whole message m. In fact, the decryptor will learn only $f(w, m)$, which is a function of the encrypted message and witness. As a result, even after many learning steps, when the actual execution of the obfuscated circuit starts, we might

aim for evaluating a ciphertext (generated during the obfuscation phase) on a *new* function. This challenge did not exist in the previous separations of Chapter 7 where we dealt with `all-or-nothing' primitives, because the probability of *not* decrypting a ciphertext during all the learning steps and then suddenly trying to decrypt it during the nal evaluation phase could be bounded to be arbitrary small. However, here we might try to decrypt this ciphertext in all these steps, every time with a di erent function, which could make the information gathered during the learning step useless for the nal evaluation.

2. **Unlearnable Hidden Queries:** To get *extended* FE from our cFWE (step (2) above), our cFWE needs to be extended as well. Namely, we allow the functions evaluated by cFWE to accept circuits with all possible gates that compute the subroutines of cFWE itself. However, for technical reasons, we limit how the witness veri cation is done in cFWE to only accept one-way function gates. Now, since we are dealing with an oracle that is an ideal version of our cFWE primitive, the function $f^{\mathsf{cFWE}}(m, w)$ may also issue queries of their own. The challenge is that there could be many such indirect/hidden queries asked during the obfuscation phase (in particular during the learning step) that we *cannot* send over to the nal evaluator simply because these queries are *not* suitable in the ideal world.

Resolving Challenges. Here we describe main ideas to resolve the challenges above.

1. To resolve the rst challenge, we add a speci c feature to cFWE so that no ciphertext c = Enc(x = (a, m)) would be decrypted more than once by the same person. More formally, we add a subroutine to FWE (as part of our cFWE) that reveals the message x = (a, m) fully, if one can provide two correct witnesses w_1 6= w_2 for the attribute a. This way, the second time that we want to decrypt c, instead we can recover the whole message x and run the function f on our own! By this trick, we will not have to worry about the fuzzy nature of FWE, as each message is now decryped at most once. In fact, adding this subroutine is the exact reason that cFWE is a *weaker* primitive than FWE.
2. To resolve the second challenge, we rely on an information theoretic argument. Suppose for simplicity that the encryption algorithm does not take an input other than the message[2] x. Suppose we use a random (injective) function Enc: x 7! c for encryption, mapping strings of length n to strings of length p = $p(n)$. Then, if p n, information theoretically, any q query algorithm who has no special advice about the oracle has a chance of q $2^{n\ p}$ to nd a valid ciphertext. If p n this probability is very small, so intuitively we would need about p n log(q) bits of advice to nd such ciphertext. On the other hand, any decryption query over a ciphertext c will only return t = $t(n)$ bits, which in our paper is assumed to be t p n. Therefore, if we interpret the decryption like a `trade' of information, we need to spend (p n) bits to get back only s $o(p$ $n)$ bits. This is the main idea behind our argument showing that during the learning phase, we will not discover more than a polynomial

[2] This is not true as the encryption is randomized, but allows us to explain the idea more easily.

number of new ciphertexts, unless we have encrypted them! By running the learning step of the compiler enough number of times, we will learn all such queries and can successfully nish the nal evaluation.

By the using above two ideas, we can successfully compile out our oracle I from any IO^I construction. The compilation process itself consists of two steps. The rst step being compiling out just the decryption queries where we face and resolve the challenges that we described above. Once we do that, we get an approximate obfuscator in a new oracle model I^0 that is actually a variant of an idealized witness encryption oracle. The second step would be to compile out the oracle I^0, which was already shown in Section 7.3, to get the desired approximate obfuscator in the plain model.

8.4 Monolithic Separation of IO from Short-Output FE

In this section, we prove our main impossibility result which states that we cannot construct an IO scheme in a monolithic way from any single-key functional encryption scheme that is restricted to handling only functions of \short" output length. More formally, we prove the following theorem.

Theorem 8.4.1. *Assume the existence of one-way functions and that* $\mathbf{NP} \not\subseteq \mathbf{co\text{-}NP}$*. Then there exists no monolithic construction of IO from any single-key* t*-bit-output functional encryption scheme where* $t(n,\kappa)$ $p(n,\kappa)$ $\omega(n+\kappa)$*,* n *is the message length,* p *is the ciphertext length, and* κ *is the security parameter of the functional encryption scheme.*

To prove Theorem 8.4.1, we will apply Lemma 7.2.7 for the idealized functional witness encryption model (formally de ned in Section 8.4.1) to prove that there is no black-box construction of IO from any primitive P that can be black-box constructed from the . In particular, we will do so for P that is the extended functional encryption primitive. Our task is thus twofold: (1) to prove that P can black-box constructed from and (2) to show a simulatable compilation procedure that compiles out from any IO construction. The rst task is proven in Section 8.4.2 and the second task is proven in Section 8.4.5. By Lemma 7.2.7, this would imply the separation result of IO from P and prove Theorem 8.4.1.

Our oracle, which is more formally de ned in Section 8.4.1, acts an idealized version of a single-key short-output functional encryption scheme, which makes the construction of secure FE quite straightforward. As a result, the main challenge lies in showing a simulatable compilation procedure for IO that satis es De nition 7.2.2 in this idealized model, and therefore, it is instructive to look at how the compilation process works and what challenges are faced with dealing with oracle .

8.4.1 The Ideal Model

In this section, we de ne the distribution of our idealized (randomized) oracle that can be used to realize some form of (extended) functional witness encryption. We also provide

several de nitions regarding the algorithms in this model and the types of queries that these algorithms can make.

Definition 8.4.2 (Randomized Functional Witness Encryption Oracle)**.** Let V be a PPT algorithm that takes as input (w, a), outputs $b \in \{0,1\}$ and runs in time poly($|a|$). Also, let F be a PPT algorithm that accepts as input a witness w and a message m then outputs a string $y \in \{0,1\}^s$. We denote the *random* $(\mathsf{V}, \mathsf{F}, p)$*-functional witness encryption* (rFWE) oracle as $\bar{\mathcal{O}}_{\mathsf{V};\mathsf{F};p} = \{\bar{\mathcal{O}}_{\mathsf{V};\mathsf{F};p;n}\}_{n\in\mathbb{N}}$ where $\bar{\mathcal{O}}_{\mathsf{V};\mathsf{F};p;n} = (\mathsf{Enc}, \mathsf{Dec}_{\mathsf{V};\mathsf{F}}, \mathsf{Rev}, \mathsf{RevMsg}_{\mathsf{V}})$ is de ned as follows:

Enc: $\{0,1\}^n \to \{0,1\}^{p(n)}$ is a random injective function mapping strings $x \in \{0,1\}^n$ to \ciphertexts" $c \in \{0,1\}^p$ where $p(n) \geq n$.

$\mathsf{Dec}_{\mathsf{V};\mathsf{F}}$: $\{0,1\}^{\ell} \to \{0,1\}^n \cup \{\bot\}$: Given $(w, c) \in \{0,1\}^{\ell}$ as input where $c \in \{0,1\}^{p(n)}$, $\mathsf{Dec}_{\mathsf{V};\mathsf{F}}(w, c)$ allows us to decrypt the ciphertext $c = \mathsf{Enc}(x)$ to get back x, parse it as $x = (a, m)$, then get $\mathsf{F}(w, m)$ as long as the predicate test is satis ed on (w, a). More formally, the following steps are performed:

1. If $\nexists\, x$ such that $\mathsf{Enc}(x) = c$, output $\bot$. Otherwise, continue to the next step.
2. Find x such that $\mathsf{Enc}(x) = c$, and parse it as $x = (a, m)$.
3. If $\mathsf{V}(w, a) = 1$, output $\mathsf{F}(w, m)$. Otherwise, output $\bot$.

Rev: $\{0,1\}^{p(n)} \to \{0,1\}^{\ell} \cup \{\bot\}$ is a function that, given an input $c \in \{0,1\}^{p(n)}$, would output the corresponding attribute $a \in \{0,1\}^{\ell}$ for which $\mathsf{Enc}((a, m)) = c$. If there is no such a then output $\bot$.

$\mathsf{RevMsg}_{\mathsf{V}}$: $\{0,1\}^{\ell'} \to \{0,1\}^{\ell} \cup \{\bot\}$: Given (w_1, w_2, c) where $w_1 \neq w_2$ and $c \in \{0,1\}^{p(n)}$, if there exist $x = (a, m)$ such that $\mathsf{Enc}(x) = c$ and $\mathsf{V}(w_i, a) = 1$ for $i \in \{1, 2\}$ then reveal m. Otherwise, output $\bot$.

When it is clear from context, we sometimes omit the subscripts from $\mathsf{Dec}_{\mathsf{V};\mathsf{F}}$, $\mathsf{RevMsg}_{\mathsf{V}}$, and $\bar{\mathcal{O}}_{\mathsf{V};\mathsf{F}}$ and simply write them as Dec, RevMsg, and $\bar{\mathcal{O}}$, respectively. Furthermore, we denote any query-answer pair (q, β) asked by some oracle algorithm A to a subroutine $T \in \{\mathsf{Enc}, \mathsf{Dec}, \mathsf{Rev}, \mathsf{RevMsg}\}$ as $(q \to \beta)_T$.

Definition 8.4.3 (Extended Randomized Functional Witness Encryption Oracle)**.** We de ne the randomized *extended* functional witness encryption oracle $\mathcal{O}_{\mathsf{V};\mathsf{F};p}$ as an rFWE oracle $\bar{\mathcal{O}}_{\mathsf{V};\mathsf{F};p} = (\mathsf{Enc}, \mathsf{Dec}_{\mathsf{V};\mathsf{F}}, \mathsf{Rev}, \mathsf{RevMsg})$ where V and F satisfy the following properties:

V is a PPT *oracle* algorithm that takes as input (w, a), interprets $a^{(\cdot)}$ as an oracle-aided circuit that can only make Enc calls, then outputs $a^{\mathsf{Enc}}(w)$.

F is a PPT *oracle* algorithm that takes as input (w, m), parses $w = (z_1, z_2)$, interprets $z_1^{(\cdot)}$ as an oracle-aided circuit that can make calls to any subroutine in $\mathcal{O} = (\mathsf{Enc}, \mathsf{Dec}, \mathsf{Rev}, \mathsf{RevMsg})$, then outputs $z_1(m)$.

For the purposes of this section, we will use the extended rFWE oracle $\mathcal{O}$ in order to prove our separation result of IO from extended functional encryption - mainly because this oracle is su cient for getting extended FE. Nevertheless, we will still make use of $\bar{\mathcal{O}}$ later on in Section 8.6 to prove the fully black-box separation of IO from (non-extended) functional encryption.

Next, we present the following de nition of canonical executions that is a property of algorithms in this ideal model. This normal form of algorithms helps us in reducing the query cases to analyze since there are useless queries whose answers can be computed without needing to ask the oracle.

Definition 8.4.4 (Canonical executions)**.** We de ne an oracle algorithm A relative to the extended rFWE oracle to be in canonical form if the following conditions are satis ed:

If A has issued a query of the form Enc(x) = c, then it will not ask $\mathsf{Dec}_{\mathsf{V};\mathsf{F}}(.,c)$, Rev($c$), or $\mathsf{RevMsg}_{\mathsf{V}}(.,.,c)$ as it can compute the answers of these queries on its own. In particular, for $\mathsf{Dec}_{\mathsf{V};\mathsf{F}}$ and $\mathsf{RevMsg}_{\mathsf{V}}$ queries, it would run V and F directly to compute the query answers correctly.

Before asking any $\mathsf{Dec}_{\mathsf{V};\mathsf{F}}(w,c)$ query where Enc(x) = c for some $x = (a,m)$, A would go through the following steps rst:

- A would get a Rev(c) then run $\mathsf{V}^{\mathsf{Enc}}(w,a)$ on its own, making sure to answer any queries of V using Enc. If $\mathsf{V}^{\mathsf{Enc}}(w,a) = 0$ then do not issue $\mathsf{Dec}_{\mathsf{V};\mathsf{F}}(w,c)$ to and use $?$ as the answer instead. Otherwise, continue to the next step.
- If A has beforehand ran $\mathsf{V}^{\mathsf{Enc}}(w',a) = 1$ for some $w' \neq w$ then it does not ask $\mathsf{Dec}_{\mathsf{V};\mathsf{F}}(w,c)$ and instead computes the answer to this query on its own. That is, it rst gets m RevMsg(w,w',c), computes on its own F (w,m) and outputs F (w,m) if $\mathsf{V}^{\mathsf{Enc}}(w,a) = 1$ or otherwise $?$.
- If A has not asked $\mathsf{Dec}_{\mathsf{V};\mathsf{F}}(w',c)$ for any $w' \neq w$ (or did but it received $?$ as the answer) then it directly asks $\mathsf{Dec}_{\mathsf{V};\mathsf{F}}(w,c)$ from the oracle.

Before asking any $\mathsf{RevMsg}_{\mathsf{V}}(w_1,w_2,c)$ query where Enc(x) = c for some $x = (a,m)$, A would go through the following steps rst:

- A would get a Rev(c) then run $\mathsf{V}^{\mathsf{Enc}}(w_i,a)$ for all $i \in \{1,2\}$ on its own, making sure to answer any queries of V using Enc. If $\mathsf{V}^{\mathsf{Enc}}(w_i,a) = 0$ for some i then do not issue $\mathsf{RevMsg}_{\mathsf{V}}(w_1,w_2,c)$ to and use $?$ as the answer instead. Otherwise, continue to the next step.
- After issuing $\mathsf{RevMsg}_{\mathsf{V}}(w_1,w_2,c)$ to and getting back an answer $m \neq ?$, ask the query Enc(x) where $x = (a,m)$ then run F (w_1,m) and F (w_2,m).

Note that any oracle algorithm A can be easily modi ed into a canonical form by increasing its query complexity by at most a polynomial factor assuming that F has extended polynomial query complexity.

Remark 8.4.5. We observe the following useful property regarding the number of queries of a speci c type that a canonical algorithm in the oracle model can make. Namely, given a canonical A, for any ciphertext c = Enc(x) where $x = (a,m)$ for which A has not asked Enc(x) before, A would ask at most one query of the form Rev(c), at most one query of the form $\mathsf{Dec}_{\mathsf{V};\mathsf{F}}(w,c)$ for which $\mathsf{V}^{\mathsf{Enc}}(w,a) = 1$, and at most one query of the form $\mathsf{RevMsg}_{\mathsf{V}}(w_1,w_2,c)$ for which $\mathsf{V}^{\mathsf{Enc}}(w_i,a) = 1$ where $i \in \{1,2\}$. Furthermore, A would never ask a query if $\mathsf{V}^{\mathsf{Enc}}(w,a) = 0$ since this condition can be veri ed independently by A and the answer can be simulated as it would invariably be $?$.

Looking ahead, we will use this property later on to prove an upper bound on the number of ciphertexts that an adversary can decrypt without knowing the underlying message. Fur-

thermore, we stress that this property holds speci cally due to the presence of the RevMsg subroutine which leaks the entire message of a given ciphertext once two di erent valid witnesses are provided. As a result, this shows that decrypting a ciphertext more than once (under di erent witnesses) does not help as the message could be revealed instead.

We also provide the following de nitions to classify the ciphertext and query types. This would simplify our discussion and clarify some aspects of the details later in the proof.

Definition 8.4.6 (Ciphertext Types)**.** Let A be a canonical algorithm in the ideal model and suppose that Q_A is the set of query-answer pairs that A asks during its execution. For any q of the form $\mathsf{Dec}_{V,F}(w, c)$, $\mathsf{Rev}(c)$, or $\mathsf{RevMsg}_V(w_1, w_2, c)$, we say that c is *valid* if there exists x such that $c = \mathsf{Enc}(x)$, and we say that c is *unknown* (to A) if the query-answer pair $(x \stackrel{?}{=} c)_{\mathsf{Enc}}$ is not in Q_A.

Definition 8.4.7 (Query Types)**.** Let A be a canonical algorithm in the ideal model and let Q_A be the query-answer pairs that it has asked so far. For any query new query q issued to , we de ne several properties that such a query might have:

Determined: We say q is determined with respect to Q_A if there exists $(q \stackrel{?}{=} \beta)_T \in Q_A$ for some answer β or there exists some query $(q' \stackrel{?}{=} \beta')_T \in Q_A$ that determines that answer of q without needing to issue q to .

Direct: We say q is a *direct* query if A issues this query to to get back some answer β. The answers to such queries are said to be *visible* to A.

Indirect: We say q is an indirect query if q is issued by F during a Dec query that was issued by A. The answers to such queries are said to be *hidden* from A.

8.4.2 Extended Functional Encryption Exists Relative to

In this section, we show how to construct a semantically-secure extended FE scheme. Namely, we prove the following:

Lemma 8.4.8. *There exists a correct and subexponentially-secure implementation of extended functional encryption in the oracle model with measure one of oracles.*

We do this in two steps: we rst show how to construct an extended variant of a functional witness encryption (for a speci c class of functions and relations) from the ideal oracle and then show how to use it to construct the desired functional encryption scheme. Our variant of FWE that we will construct is de ned as follows.

Definition 8.4.9 (Customized Functional Witness Encryption (CFWE))**.** Given any one-way function R, let V be a PPT oracle algorithm that takes as input an instance-message pair $x = (a, m)$ and witness w, interprets a as a Boolean oracle circuit that can only make calls to R (it returns 0 otherwise) then outputs $a^R(w)$. Furthermore, let F be a PPT oracle algorithm that accepts as input a string $w = (z_1, z_2)$ and a message m, interprets z_1 as a circuit then outputs a string $y = z_1(m)$.

For any given security parameter κ, a *customized functional witness encryption* scheme parameterized with V and F consists of three PPT algorithms P = (Enc, $\mathsf{Dec}_{\mathsf{V};\mathsf{F}}$, Rev) de ned as follows:

Enc(1 $^{\kappa}$, a, m) : given an instance $a \in \{0,1\}^{\ell}$, message $m \in \{0,1\}^{\ell}$, and security parameter κ, outputs $c \in \{0,1\}^{*}$.

Rev(c) : given a ciphertext c, outputs the corresponding attribute a under which the message is encrypted.

$\mathsf{Dec}_{\mathsf{V};\mathsf{F}}(w, c)$: given ciphertext c and \witness" string $w \in \{0,1\}^{*}$, outputs a message $m' \in \{0,1\}^{*}$.

A customized functional witness encryption scheme satis es the following completeness and security properties:

Correctness: For any security parameter κ, any $m \in \{0,1\}^{\ell}$, and any $(w, (a, m))$ such that $\mathsf{V}^{R}(w, a) = 1$, it holds that

$$\Pr_{\mathsf{Enc};\mathsf{Dec}}[\mathsf{Dec}_{\mathsf{V};\mathsf{F}}(w, \mathsf{Enc}(1^{\kappa}, a, m)) = \mathsf{F}^{P}(w, m)] = 1$$

Instance-Revealing: For any security parameter κ, any $m \in \{0,1\}^{\ell}$, and any $(w, (a, m))$ such that $\mathsf{V}^{R}(w, a) = 1$, it holds that

$$\Pr[\mathsf{Rev}(\mathsf{Enc}(1^{\kappa}, a, m)) = a] = 1$$

Weak Extractability: For any PPT adversary A and polynomial $p_1(.)$, there exists a PPT extractor E and a polynomial $p_2(.)$ such that for any security parameter κ, any a for which $\mathsf{V}^{R}(w, a) = 1$ for some w, and any m_0, m_1 where $|m_0| = |m_1|$, if:

$$\Pr\left[A(1^{\kappa}, c) = b \mid b \xleftarrow{\$} \{0,1\}, c \leftarrow \mathsf{Enc}(1^{\kappa}, a, m_b)\right] \geq \frac{1}{2} + \frac{1}{p_1(\kappa)}$$

Then:

$$\Pr\left[\begin{array}{c} E^{A}(1^{\kappa}, a, m_0, m_1) = w : \mathsf{V}^{R}(w, a) = 1 \wedge \mathsf{F}^{P}(w, m_0) \neq \mathsf{F}^{P}(w, m_1) \\ E^{A}(1^{\kappa}, a, m_0, m_1) = (w_1, w_2) : w_1 \neq w_2 \wedge \mathsf{V}^{R}(w_1, a) = 1 \wedge \mathsf{V}^{R}(w_2, a) = 1 \end{array}\right] \geq \frac{1}{p_2(\kappa)}$$

Note that we declared such a functional witness encryption as being customized since it deviates slightly (in an incomparable way) to standard FWE (De nition 2.6.10). Particularly, this customized FWE is essentially an *extended* FWE scheme with a *weaker* extractability property and is de ned for a speci c V and a speci c F de ned above. However, in order to minimize the number of descriptors attached to this primitive, we will simply say that this FWE is customized with the properties listed in De nition 8.4.9. It is important to note however that, irrespective of what this primitive is called or how it is de ned, what we eventually care about is that one can use it to get extended FE, which is our desired target primitive.

8.4.3 Customized FWE in the Ideal Model

Here we provide the construction of customized FWE using the $_{V;F}$ oracle. We note that can be thought of as an ideal customized FWE and hence the construction of the CFWE primitive is straightforward.

Construction 8.4.10 (Customized Functional Witness Encryption)**.** Let V and F be as de ned in De nition 8.4.9. For any security parameter κ and oracle $_{V;F}$ sampled according to De nition 8.4.3, we will implement a customized FWE scheme P de ned by V and function class F as follows:

CFWE.Enc(1 $, a, m$) : Given a 2 $f0,1g$, message m 2 $f0,1g^{n'}$ and security parameter 1 , let n = $(n' + jaj + \kappa)$. Sample r $f0,1g$ uniformly at random then output c = Enc(x) where $x = (a,(m,r))$.

CFWE.Dec(w, c) : Given string w and ciphertext c 2 $f0,1g^p$, get y $\text{Dec}_{V;F}(w,c)$, then output y.

CFWE.Rev(c) : Given ciphertext c 2 $f0,1g^p$, outputs Rev(c).

Lemma 8.4.11. *Construction 8.4.10 is a correct and subexponentially-secure implementation of customized functional witness encryption in the oracle model with measure one.*

The correctness of the scheme follows immediately. The security also holds intuitively, since if there is any secure realization of this primitive, it should be the idealized version of it de ned as an oracle. To prove the security of this construction formally, we will show that if there exists an adversary A against scheme P (in the $_{V;F}$ oracle model) that can distinguish between encryptions of two di erent messages m_0, m_1 (under the same instance a) with non-negligible advantage then there exists a (xed) deterministic straight-line extractor E with access to $_{V;F}$ that can nd the valid witness w for the underlying instance a of the challenge ciphertext such that F(w $, m_0$) 6= F(w $, m_1$) or nd two distinct witnesses (w_1, w_2) such that V(w_1, a) = V(w_2, a) = 1.

Suppose A is an adversary in the indistinguishability game with success probability $1/2 + \epsilon$. Then there exists a canonical adversary D that satis es De nition 8.4.4 and has the same success probability as A. Let Q_E be the query-answer pairs asked by the extractor during its execution. The extractor E would work as follows: given (a, m_0, m_1) as input and acting as the challenger for adversary D, it runs D (1 $, c$) where c Enc(1 $, a, m_b$) is the challenge ciphertext and b $\overset{\$}{}$ $f0,1g$. To answer any query q asked by D, the extractor will emulate the answer returned to A as follows:

If q is a query of the form Enc(x), it forwards the query to the oracle to get some answer c, adds $(x \,?\, c)_{\text{Enc}}$ to Q_E then returns c to A.

If q is a query of the form Rev(c) for c 6= c then it returns ?[3]. If $c = c$ then E returns a as the answer.

If q is a query of the form $\text{Dec}_{V;F}(w,c)$ for c 6= c then the extractor aborts with ?. If $c = c$ then it must be the case that Dec(w, c) 6= ? (otherwise the canonical D would

[3]Even if it is a valid ciphertext - however querying an unknown valid ciphertext happens with negligible probability

not have asked it). Then we have two possibilities: $\mathsf{F}(w, m_0) = \mathsf{F}(w, m_1) = y$ in which case D can compute the answer y (correctly) on its own without issuing this query. The other possibility would be $\mathsf{F}(w, m_0) \neq \mathsf{F}(w, m_1)$ in which case E halts execution and outputs w as its answer.

If q is a query of the form $\mathsf{RevMsg}_{\mathsf{V}}(w_1, w_2, c)$ for $c \neq c$ then the extractor aborts with $?$. If $c = c$ then it must be the case that $\mathsf{RevMsg}_{\mathsf{V}}(w_1, w_2, c) \neq ?$ (otherwise the canonical D would not have asked it). In that case, E halts execution and outputs (w_1, w_2).

If D has completed execution and E has not observed a query that led it to halt and output a witness (or a pair of witnesses), the extractor will abort with failure and output $?$.

Lemma 8.4.12. *For any PPT canonical oracle adversary D against Construction 8.4.10, any instance a, and any $m_0 \neq m_1$ of the same length $|m_0| = |m_1|$, if there exists a non-negligible function $\epsilon(.)$ such that:*

$$\Pr\left[D\ (1^\kappa, c) = b \mid b \xleftarrow{\$} \{0,1\}, c \leftarrow \mathsf{Enc}(1^\kappa, a, m_b)\right] \geq \frac{1}{2} + \epsilon(\kappa) \tag{8.1}$$

Then there exists a PPT straight-line extractor E, a non-negligible function $\epsilon'(.)$ and a negligible function negl(.) *such that:*

$$\Pr\left[\begin{array}{l} E^{\langle D, \cdot\rangle}(a, m_0, m_1) = w \ \wedge\ \mathsf{V}^{\mathsf{Enc}}(w, a) = 1 \wedge \mathsf{F}(w, m_0) \neq \mathsf{F}(w, m_1) \\ \\ E^{\langle D, \cdot\rangle}(a, m_0, m_1) = (w_1, w_2) \wedge \overline{\mathsf{V}}^{\mathsf{Enc}}(w_1, a) = 1 \wedge \mathsf{V}^{\mathsf{Enc}}(w_2, a) = 1 \end{array}\right] \geq \epsilon'(\kappa) \geq \mathsf{negl}(\kappa) \tag{8.2}$$

Proof. Let D be an adversary satisfying Equation (8.1) above. We first define the following events that will be of interest to us:

Let AdvWin be the event that D succeeds in the distinguishing game of Equation (8.1).

Let ExtWin be the event that the extractor succeeds in extracting a witness or a pair of witnesses (as in Equation (8.2) above).

Let Bad to be the event that D asks (directly or indirectly via F) a query of the form $\mathsf{Dec}_{\mathsf{V},\mathsf{F}}(w, c')$ or $\mathsf{RevMsg}_{\mathsf{V}}(w_1, w_2, c')$ for some $c' \neq c$ for which it has not asked $\mathsf{Enc}(x) = c'$ previously.

Note that as long as Bad does not happen, the extractor will not abort due to failing to answer a valid D Dec or RevMsg query. Observe that by a union bound:

$$\Pr_{\kappa,b,r}[\overline{\mathsf{ExtWin}}] \leq \Pr_{\kappa,b,r}[\overline{\mathsf{ExtWin}} \wedge \mathsf{AdvWin} \wedge \overline{\mathsf{Bad}}] + \Pr_{\kappa,b,r}[\overline{\mathsf{AdvWin}}] + \Pr_{\kappa,b,r}[\mathsf{Bad}]$$

where r is the randomness of the distinguisher D.

Note that, over the randomness of κ, we find that $\Pr_{b,r}[\mathsf{Bad}] \leq \mathsf{negl}(\kappa)$ by a standard argument that evaluate the oracle in a `lazy' way (upon request) as this is the probability of hitting a point in the image of a random injective function without knowing the preimage. Furthermore, since $\Pr[\mathsf{AdvWin}] \geq 1/2 + \epsilon$ for some non-negligible function ϵ, it suffices to show

that Pr[$\overline{\text{ExtWin}}$ ^ AdvWin ^ $\overline{\text{Bad}}$] is negligible. Note that, by our construction of extractor E, this event is equivalent to saying that the adversary D succeeds in the distinguishing game but never asks a query of the form $\text{Dec}_{V,F}(w, c)$ $\notin$? for which $F(w, m_0)$ $\notin$ $F(w, m_1)$ or a query of the form $\text{RevMsg}_V(w_1, w_2, c)$ $\notin$? and so the extractor fails to recover the witness(es). For simplicity of notation de ne Win := $\overline{\text{ExtWin}}$ ^ AdvWin ^ $\overline{\text{Bad}}$.

We will show that, with overwhelming probability over the choice of oracle , the probability of Win happening is only a negligible factor over the trivial advantage. That is, we will prove the following lemma:

Lemma 8.4.13. *For security parameter κ and $\epsilon > 2^{-10}$,* Pr [$\Pr_{b,r}$[Win] $1/2 + \epsilon$] $2^{-2^{2\kappa/5}}$

Proof. Since Bad does not happen and D wins but E does not, we can conclude that D only issues encryptions queries to . We proceed to show that D only succeeds with negligible advantage. De ne Hit to be the event that D happens to ask $\text{Enc}(x) = c$. Then we have that:

$$\Pr\left[\Pr_{b,r}[\text{Win}] \quad \frac{1}{2} + \epsilon\right] \quad \Pr\left[\Pr_{b,r}[\text{Win} \wedge \overline{\text{Hit}}] + \Pr_{b,r}[\text{Hit}] \quad \frac{1}{2} + \epsilon\right]$$
$$\Pr\left[\Pr_{b,r}[\text{Win} \wedge \overline{\text{Hit}}] \quad \frac{1}{2} + \frac{\epsilon}{2} - \Pr_{b,r}[\text{Hit}] \quad \frac{\epsilon}{2}\right]$$
$$\Pr\left[\Pr_{b,r}[\text{Win} \wedge \overline{\text{Hit}}] \quad \frac{1}{2} + \frac{\epsilon}{2}\right] + \Pr\left[\Pr_{b,r}[\text{Hit}] \quad \frac{\epsilon}{2}\right]$$

We can bound the event Hit from happening since it is the event that we invert the image of a random injective function. This can be done over the randomness of the oracle then, using an averaging argument, deduce that the probability that Hit happens for a non-negligible fraction of oracles is negligible.

We will thus focus on proving the rst term. In doing so, we will reduce the problem of indistinguishability to that of predicting the output of a random Boolean function on a random point in the domain of this function. We then prove that the latter problem is hard using a compression technique which allows to argue that an adversary that can win the prediction game with non-negligible advantage can only do so for a negligible fraction of the oracles. We will make use of the following lemma that shows the existence of a randomized compression technique for random Boolean functions using adversaries against the prediction game.

Lemma 8.4.14 (Fact 10.1 and Lemma 10.4 [DTT10])**.** *Let A be an oracle algorithm that makes q queries to some xed oracle predicate $p : \{0,1\}^n \to \{0,1\}$, never queries the oracle on its input and satis es the following for some ϵ:*

$$\Pr[A^p(x) = p(x)] \quad 1/2 + \epsilon$$

Then there exists a randomized encoder E and a randomized decoder D such that:

$$\Pr_r[D(E(p, r), r) = p] \quad \epsilon/q$$

and $|E(p,r)|$ 2^n $\epsilon^2 2^n/q$.

For any given adversary D that wins in the indistinguishability game of FWE with probability 1/2 + ϵ/2 without asking any Dec or RevMsg queries, we can construct a new oracle-aided adversary $\widetilde{D}$ that aims to win in the experiment $\mathsf{Exp}^{P}_{\widetilde{D}}(1^\kappa)$ as defined in Figure 8.1. The adversary $\widetilde{D}$ will have access to oracle and P, which is defined as follows:

$$P(a,r,c_0,c_1) = \begin{cases} 0 \text{ if } c_0 = \mathsf{Enc}(a|0|r) \text{ and } c_1 = \mathsf{Enc}(a|1|r) \\ 1 \text{ if } c_0 = \mathsf{Enc}(a|1|r) \text{ and } c_1 = \mathsf{Enc}(a|0|r) \\ ? \text{ otherwise} \end{cases}$$

However, we note that $\widetilde{D}$ will not query P on its input based on the definition of the experiment and will not ask any Dec or RevMsg queries from since D will not either.

Experiment $\mathsf{Exp}^{P}_{A}(1^\kappa, a)$:

1. $r \xleftarrow{\$} \{0,1\}^\kappa, b \xleftarrow{\$} \{0,1\}$
2. $c_0 \leftarrow \mathsf{Enc}(a|b|r), c_1 \leftarrow \mathsf{Enc}(a|(1-b)|r)$
3. $b' \leftarrow A^{P(\cdot)}(1^\kappa, r, c_0, c_1)$ where A is not allowed to query P on its input
4. Output 1 if $b = b'$ and 0 otherwise.

Figure 8.1: The single-instance Exp^{P}_{A} Experiment

The adversary $\widetilde{D}$, given $(1^\kappa, r, c_0, c_1)$, would execute $b' \leftarrow D(1^\kappa, c_0)$ and output b' as its answer. We can modify $\widetilde{D}$ so that whenever it issues a query to $\mathsf{Enc}(a|b|r) = c_b$, it would also call $\mathsf{Enc}(a|1-b|r) = c_{1-b}$ followed by a call to $P(a,r,c_0,c_1)$. This ensures that any encryption query issued to is translated into a query to P.

Define $P_a := P(a,.,.,.)$ to be the random predicate P restricted to attribute a where $P_a : \{0,1\}^\ell \to \{0,1\}$ and $\ell = \kappa + 2|c|$. Note that $P = \{P_a\}_{a \in \mathsf{N}}$ can be interpreted as an alternative description for . That is, given P, one can reconstruct on any point in its domain. By Lemma 8.4.14, we can encode P_a using at most $2^\ell - \epsilon^2 2^\ell/4q$ bits where q is the number of queries that $\widetilde{D}$ makes. Thus, we have compressed the entire oracle P by saving $\alpha = \epsilon^2 2^\ell/4q$ bits. Hence, assuming that ϵ $\mathsf{negl}(\kappa)$ and q $\mathsf{poly}(\kappa)$ we find that the fraction of oracles for which $\widetilde{D}$ can win on is at most $1/2$ + $\mathsf{negl}(\kappa)$ assuming that we encrypt attributes a of size $\mathsf{poly}(\kappa)$ which would imply that $\ell = \mathsf{poly}(\kappa)$. □

To conclude the proof of Lemma 8.4.12, we note that by Lemma 8.4.13 and the fact that Pr[Bad] is $\mathsf{negl}(\kappa)$ we find that $\Pr_{;b;r}[\overline{\mathsf{ExtWin}}]$ $\mathsf{negl}(\kappa)$. □

Proof of Lemma 8.4.11. It is clear that the Construction 8.4.10 is correct. Furthermore, by Lemma 8.4.12, it also satis es the extractability property. □

8.4.4 From CFWE to Functional Encryption

Construction 8.4.15 (Functional Encryption)**.** Let P_F = (FE. Setup, FE. Keygen, FE. Enc, FE. Dec) be the functional encryption scheme for the function family F that we would like to construct. Suppose Sig = (Sig.Gen, Sig.Sign, Sig.Ver) is a secure signature scheme.

De ne a language L with an associated PPT veri er V such that an instance a of the language corresponds to the signature veri cation circuit Sig.Ver(vk, .) that takes as input w = (f, sk$_f$) so that V(w, a) = $a(w)$ = 1 if and only if Sig.Ver(vk, w) = 1 for some oracle-aided f *2* F, sk$_f$ Sig.Sign(sk, f), and (sk, vk) Sig.Gen(1). Furthermore, let F^0 be a PPT algorithm that takes as input w = (f, sk$_f$) and a message m then outputs y = F^0(w, m) = $f(m)$.

Given a customized functional witness encryption scheme CFWE = (CFWE.Enc, CFWE.Dec$_{V;F^0}$, CFWE.Rev) for V and F^0 de ned above, signature scheme Sig, and security parameter κ, we implement the extended FE scheme P_F as follows:

- FE. Setup(1) : Generate (sk, vk) Sig.Gen(1). Output (MPK, MSK) where MPK = vk and MSK = sk.
- FE. Keygen(MSK, f) : Given MSK = sk and f *2* F, output SK$_f$ = (f, sk_f) where sk_f Sig.Sign(MSK, f).
- FE. Enc(MPK, m) : Given MPK *2* $f0,1g$ and message m *2* $f0,1g^{n'}$, output ciphertext c = CFWE.Enc(1 , MPK, m).
- FE. Dec(SK$_f$, c) : Given SK$_f$ = (f, sk_f) and ciphertext c *2* $f0,1g^p$, call and output the value returned by CFWE.Dec$_{V;F^0}$(SK$_f$, c).

Lemma 8.4.16. *Construction 8.4.15 is a fully black-box construction of extended functional encryption from customized functional witness encryption.*

Proof. We rst show that the construction is correct. Given (MPK, MSK) FE. Setup(1), for any encryption c FE. Enc(MPK, m) of a message m *2* $f0,1g^{n'}$ and functional decryption key SK$_f$ FE. Keygen(MSK, f) for a function f *2* F, we get that, if V(w, a) = $a^{Sig}(w)$ = Sig.Ver(vk, (f, sk_f)) = 1 then:

$$\text{FE. Dec}(\text{SK}_f, c) = \text{CFWE.Dec}_{V;F^0}((f, sk_f), c) = F^0((f, sk_f), m) = f^{P_F}(m)$$

Note that, since this is an monolithic construction, f can have oracle gates to any subroutine in P_F. As a result, we need to make sure that V are F^0 are speci ed in a way so that all monolithic computations are valid. First, V only has one Sig.Ver gate which is supported by OWFs. Furthermore, F^0 calls f which has oracle gates to any subroutine in P_F. Nevertheless, we can reduce each gate to P_F to CFWE or OWF gates. In particular, FE. Setup can be reduced to Sig.Gen gates, FE. Keygen can be reduced to Sig.Sign gates, FE. Enc can be reduced to CFWE.Enc gates, and FE. Dec can be reduced to CFWE.Dec gates. Thus, all gates in F^0 can be reduced to those in FWE or one-way functions.

Next, we prove the security of the scheme by reducing it to the underlying security of CFWE and Sig. Let A be a computationally bounded adversary that asks one functional secret key query and breaks the security of the FE scheme. That is, for some non-negligible $\epsilon(.)$:

$$\Pr[\mathsf{IND}_A^{1\mathsf{FE}}(1^\kappa) = 1] \quad \frac{1}{2} + \epsilon(\kappa)$$

where $\mathsf{IND}_A^{1\mathsf{FE}}$ is the experiment of De nition 2.6.9.

Towards contradiction, we will now show that, given A, we can build an attacker B that can break the strong existential unforgeability of the signature scheme under chosen message attack. On receiving the public-key MPK from the (signature game) challenger, B forwards MPK to A and upon receiving (f, m_0, m_1), requests the signature for f and then randomly chooses a message to encrypt. Note that, since FE.Enc(MPK, m_b) = CFWE.Enc(1^λ, MPK, m_b), B can use A to build a distinguisher A' against CFWE. B then runs the black-box straight-line extractor $E^{A'}$ (guaranteed to exist by the security de nition of CFWE) where at least one of the following events will happen with non-negligible probability:

The extractor returns a single witness $w = (f, sk_f)$ such that V(w, MPK) outputs 1 and $\mathsf{F}'(w, m_0) \neq \mathsf{F}'(w, m_1) \Rightarrow f(m_0) \neq f(m_1)$. Note that this implies that sk_f is a valid forgery since f cannot be the function f that A requests the signature for (because $f(m_0) = f(m_1)$ in that case) and w passed veri cation thus violating the security of the signature scheme.

The extractor returns a pair of witnesses (w_1, w_2) such that $w_1 \neq w_2$ and V(w_1, MPK) = V(w_2, MPK) = 1. This either implies that $w_i = (f, sk_f)$ for some $i \in \{1, 2\}$ is a valid witness and $f \neq f$ in which case we have a signature forgery, or it implies that $w_i = (f, sk_f')$ for some $i \in \{1, 2\}$ and hence $sk_f' \neq sk_f$ (since even if $w_{i-1} = (f, sk_f)$ we have that $w_i \neq w_{i-1}$) which is also signature forgery.

In both of the above cases, an attack against the FE scheme results in an attack against the underlying signature scheme.

□

8.4.5 Compiling out from IO

In this section, we show a simulatable compiler for compiling out $_{\mathsf{V},\mathsf{F}}$ when F is short-output. We adapt the approach outlined in Section 7.2.1 to the extended rFWE oracle $_{\mathsf{V},\mathsf{F}} = (\mathsf{Enc}, \mathsf{Dec}_{\mathsf{V},\mathsf{F}}, \mathsf{Rev}, \mathsf{RevMsg}_{\mathsf{V}})$ while making use of Lemma 7.2.7, which allows us to compile out $_{\mathsf{V},\mathsf{F}}$ in two phases: we rst compile out part of $_{\mathsf{V},\mathsf{F}}$ to get an approximately-correct obfuscator $\widehat{O}$ in the random instance-revealing witness encryption model (that produces an obfuscation $\widehat{B}$ in the -model), and then use the previous result of Section 7.3 to compile out and get an obfuscator O' in the plain-model. Since we are applying this lemma only a constant number of times, security should still be preserved. Speci cally, we will prove the following lemma:

Lemma 8.4.17. *Let* F *be a PPT oracle Turing machine that accepts as input a witness* w *and a message* m *then outputs a string* y *2* $f0,1g^s$ *where* $s(n)$ $t(n)$*. Let be a random instance-revealing witness encryption oracle. Then for any* $_{V;F;p}$ *satisfying* $t(n)$ $p(n)$ $\omega(n)$ *and for* v $_{V;F;p}$*, the following holds:*

For any IO in the $_{V;F;p}$ *ideal model, there exists a simulatable compiler with correctness error* $\epsilon < 1/200$ *for it that outputs a new obfuscator in the random instance-revealing witness encryption oracle model.*

For any IO in the oracle model, there exists a simulatable compiler with correctness error $\epsilon < 1/200$ *for it that outputs a new obfuscator in the plain model.*

We observe that by compiling out only the Dec queries of , we will end up with queries only to Enc, Rev, and RevMsg. However, we note that Enc and Rev already are part of and RevMsg can in fact be interpreted as the decryption subroutine of where $w' = (w_1, w_2)$ is de ned as the witness to the decryption subroutine. Therefore, the second part of Lemma 8.4.17 follows directly from Lemma 7.3.13, which implies that we can compile out the ideal witness encryption oracle from any IO scheme, and thus we focus on proving the rst part of the lemma. We will present the construction of the obfuscator in the random instance-revealing witness encryption model that, given an obfuscator in the model, would compile out and emulate queries to Dec, while forwarding any Enc, Rev, RevMsg queries to . Throughout this section, for simplicity of notation, we will denote = $_{V;F;p}$ to be the oracle satisfying $t(n)$ $p(n)$ $\omega(n)$.

Remark 8.4.18. For simplicity of exposition, we assume that the compiler only asks the oracle for queries from $_n$. However, our argument directly extends to handle arbitrary calls to the oracle using the following standard technique. As we will show, the \error" in our poly-query compiler in the ideal model will be at most poly$(q)/2^n$ (where q = poly(κ) is a xed polynomial over the security parameter κ of the IO construction) when we only call $_n$. It is also the case that this error adds up when we work with several input lengths $n_1, n_2, \ldots$, but it is still bounded by union bound. Therefore, the total error of the transformation will be at most O(poly$(n_1)/2^{n_1}$) where n_1 is the smallest integer for which $_{n_1}$ is queried at some point. To make n_1 large enough (to keep the error small enough) we can modify all the parties to query on *all* oracle queries up to input parameter $n_1 = c(\log(\kappa))$ for su ciently large c. (Note that this will be a polynomial number of queres in total.)

The new obfuscator $\widehat{O}$ in the instance-revealing witness encryption model

Given a δ-approximate obfuscator O = (iO, Ev) in the rFWE oracle model, we construct an (δ + ϵ)-approximate obfuscator $\widehat{O}$ = ($\widehat{\text{iO}}, \widehat{\text{Ev}}$) in the oracle model. Throughout this process, we can assume that iO and Ev are in their canonical form as in De nition 8.4.4.

Algorithm 4: `EmulateCall`

Input: Query-answer set Q, query q to a subroutine of T 2 fEnc, Dec, Rev, RevMsg$g$ of

Oracle: Random Instance-Revealing Witness Encryption Oracle = (WEnc, WDec, WRevAtt)

Output: A query-answer pair ρ_q, and the set W of hidden queries

Begin:

if 9 $(q$ 7! $\beta)_T$ 2 Q for some answer β **then**
  Set ρ_q = $(q$ 7! $\beta)_T$
end
if q = x is a query to Enc **then**
  Set ρ_q = $(x$ 7! WEnc(x))$_{\text{Enc}}$
end
if q = c is a query to Rev **then**
  Set ρ_q = $(c$ 7! WRevAtt(c))$_{\text{Enc}}$
end
if q = (w_1, w_2, c) is a query to RevMsg$_{\text{V}}$ **then**
  Set ρ_q = $(x$ 7! WDec$_{\text{V}'}((w_1, w_2), c))_{\text{Enc}}$
end
/* We simulate Dec queries */
if q = (w, c) is a query to Dec$_{\text{V;F}}$ **then**
  Let a_R be the attribute returned by `EmulateCall`(Q, q_R) where q_R is the query Rev(c)
  Emulate b V$^{\text{Enc}}(w, a_R)$ while emulating any queries using `EmulateCall`
  if b = 1 and 9 $((a, m)$ 7! $c)_{\text{Enc}}$ 2 Q **then**
    Emulate y F (w, m) while simulating any queries using `EmulateCall`
    Set W to be the set of query-answer pairs asked by F
    Set ρ_q = $((w, c)$ 7! $y)_{\text{Dec}}$
  else
    Set ρ_q = $((w, c)$ 7! $?)_{\text{Dec}}$
  end
end
Return (ρ_q, W)

Subroutine $\widehat{\mathsf{iO}}$ (C):

1. *Emulation phase*: Emulate iO (C). Initialize Q_O = ? to be the set of query-answer pairs asked by the obfuscation algorithm iO. For every query q asked by iO (C), call (ρ_q, W) EmulateCall (Q_O, q) and add ρ_q to Q_O.

2. *Learning phase*: Set Q_B = ? to be the set of direct (visible) query-answer pairs asked during this phase (so far) and Q_B^h = ? to be the set of indirect (hidden) query-answer pairs (see De nition 8.4.7). Let k = $(\ell_O + \kappa)/\epsilon$ where ℓ_O *j*iO*j* represents the number of queries asked by iO. Choose λ $\overset{\$}{}$ $[k]$ uniformly at random then for i = *f*1, ..., λ*g* do the following:

   Choose z_i $\overset{\$}{}$ *f*0, 1*g*^{jCj} uniformly at random
   Run Ev (B, z_i). For every query q asked by Ev (B, z_i), run the following and get (ρ_q, W) EmulateCall $(Q_O$ *[* Q_B *[* $Q_B^h, q)$, then add ρ_q to Q_B and W to Q_B^h.

3. The output of the -model obfuscation algorithm $\widehat{\mathsf{iO}}$ (C) will be $\widehat{B}$ = (B, Q_B).

Subroutine $\widehat{\mathsf{Ev}}$ $(\widehat{B}, z)$: Initialize $Q_{\widehat{B}}$ = ? to be the set of queries asked when evaluating $\widehat{B}$. To evaluate $\widehat{B}$ = (B, Q_B) on a new random input z we simply emulate Ev (B, z) as follows. For every query q asked by Ev (B, z), run and set (ρ_q, W) = EmulateCall $(Q_B$ *[* $Q_{\widehat{B}}, q)$ then add $(\rho_q$ *[* $W)$ to $Q_{\widehat{B}}$.

The running time of $\widehat{\mathsf{iO}}$. We note that the running time of the new obfuscator $\widehat{\mathsf{iO}}$ remains polynomial time since we are emulating the original obfuscation once followed by a polynomial number λ of learning iterations. Furthermore, since we are working with the extended oracle (see De nition 8.4.3), the way that F is de ned (as a universal circuit evaluator) makes it so that the number of recursive calls that appear due to emulating F is upper-bounded by some polynomial (in fact even quadratic).

Proving Approximate Correctness. De ne $Q_{\widehat{B}}^h$ to be the set of hidden queries asked during the nal execution phase. Set Q_T = Q_O *[* Q_B *[* Q_B^h *[* $Q_{\widehat{B}}$ *[* $Q_{\widehat{B}}^h$ to be the set of all (visible and hidden) query-answer pairs asked during all the phases. We consider two distinct experiments that construct the oracle model obfuscator exactly as described above but di er when evaluating $\widehat{B}$:

Real Experiment: $\widehat{Ev}$ $(\widehat{B}, z)$ emulates Ev (B, z) on a random input z and answers any queries using EmulateCall.

Ideal Experiment: $\widehat{\mathsf{Ev}}$ $(\widehat{B}, z)$ executes Ev (B, z) and answers all the queries of Ev (B, z) using the actual oracle .

Note that the actual emulation of the new obfuscator is statistically close to an ideal emulation of the obfuscation and learning phases using and so it su ces to compare only the real and ideal nal execution phases. In essence, in the real experiment, we can think of the

execution as Ev$^{\widehat{}}(B, z)$ where $\widehat{}$ is the oracle simulated using the learned query-answer pairs Q_B and oracle . We will compare the real experiment with the ideal experiment and show that the statistical distance between these two executions is at most ϵ. In order to achieve this, we will identify the events that make the executions Ev (B, z) and Ev$^{\widehat{}}(B, z)$ diverge (i.e. without them happening, they proceed statistically the same).

Let q be a new query that is being asked by Ev$^{\widehat{}}(B, z)$ (i.e. in the real experiment) and handled using `EmulateCall` $(Q_B \ [\ Q_{\widehat{B}}, q)$. The following are the cases that should be handled:

1. If q is a query of type Enc(x), then the answer to q will be distributed the same in both experiments as they will be both answered using the subroutine WEnc(c) of .

2. If q is a query of type Rev(c), then the answer to q will be distributed the same in both experiments as they will be both answered using the subroutine WRevAtt(c) of .

3. If q is a query of type RevMsg$_V(w_1, w_2, c)$, then the answer to q will be distributed the same in both experiments as they will be both answered using the subroutine WDec$_{V^0}(w^0, c)$ where $w^0 = (w_1, w_2)$.

4. If q is a query of type Dec$_{V,F}(w, c)$ whose answer is determined by $Q_B \ [\ Q_{\widehat{B}}$ in the real experiment then it is also determined by Q_T $(Q_B \ [\ Q_{\widehat{B}})$ in the ideal experiment and the answers are therefore distributed the same.

5. Suppose q is a query of type Dec$_{V,F}(w, c)$ that is not determined by $Q_B \ [\ Q_{\widehat{B}}$ in the real experiment. Then the answer returned by `EmulateCall` is *?* since the underlying encryption query $((a, m) \stackrel{?}{=} c)_{\text{Enc}}$ is not known. In that case, we have to consider three di erent counterparts in the ideal experiment:

   (a) **Bad Event 1:** If q is not determined by Q_T in the ideal experiment then this implies that the ideal execution Ev (B, z) is for the rst time hitting a valid ciphertext that was never generated by an encryption query asked during any of the phases. In that case, since Enc is injective, the answer returned by would be *?* with overwhelming probability.

   (b) **Bad Event 2:** The query q is determined by $Q_T \ n \ (Q_B \ [\ Q_{\widehat{B}})$ in the ideal experiment and the ideal execution Ev (B, z) has hit a valid unknown ciphertext that was generated by an encryption query in the obfuscation phase that was never learned. In this case, the answer will be F (w, m) if the veri cation passes and *?* otherwise.

   (c) **Bad Event 3:** The query q is determined by $Q_T \ n \ (Q_B \ [\ Q_{\widehat{B}})$ in the ideal experiment then and the ideal execution Ev (B, z) has hit a valid unknown ciphertext that was generated as a hidden query (i.e. issued by inner F executions) during the learning or evaluation phases. In this case, the answer will be F (w, m) if the veri cation passes and *?* otherwise.

Notice that the answer to such a query in the ideal experiment di ers from that in the real experiment (which always outputs ?). However, we will show below that such an event is unlikely to occur.

For circuit input z, let $E(z)$ be the event that either one of Cases 5a, 5b, or 5c happen. More speci cally, this is the event that $\widehat{\text{Ev}}(B, z)$ asks a query q of the form $\text{Dec}_{\text{V;F}}(w, c)$ where c is a valid ciphertext that was either (i) never generated before during any of the phases, (ii) generated during the obfuscation phase, or (iii) generated by a hidden query in the learning and/or nal evaluation phases. Assuming that event $E(z)$ does not happen, both experiments will proceed identically the same and the output distributions of $\text{Ev}(B, z)$ and $\widehat{\text{Ev}}(B, z)$ will be statistically close. More formally, the probability of correctness for $\widehat{iO}$ is:

$$\Pr_z[\widehat{\text{Ev}}(B, z) \not\in C(z)] = \Pr_z[\widehat{\text{Ev}}(B, z) \not\in C(z) \wedge \neg E(z)] + \Pr_z[\widehat{\text{Ev}}(B, z) \not\in C(z) \wedge E(z)]$$
$$\Pr_z[\widehat{\text{Ev}}(B, z) \not\in C(z) \wedge \neg E(z)] + \Pr_z[E(z)]$$

By the approximate functionality of iO, we have that:

$$\Pr_z[\text{iO}(C)(z) \not\in C(z)] = \Pr_z[\text{Ev}(B, z) \not\in C(z)] \quad \delta(\kappa)$$

Therefore,

$$\Pr_z[\widehat{\text{Ev}}(B, z) \not\in C(z) \wedge \neg E(z)] = \Pr_z[\text{Ev}(B, z) \not\in C(z) \wedge \neg E(z)] \quad \delta \tag{8.3}$$

We are thus left to show that $\Pr[E(z)] \quad \epsilon$. Since both experiments proceed the same up until E happens, the probability of E happening is the same in both worlds and we will thus choose to bound this bad event in the ideal world.

Proof Intuition. At a high-level, in order to show that E is unlikely, we will show that the learning procedure and nal execution phases, when treated as a single non-uniform query-adaptive algorithm A, will only ask a bounded number of queries for valid ciphertexts whose corresponding underlying message is unknown to this algorithm. Then, given this upper bound on such queries, we ensure that by running the learning procedure for su cient number of times, the nal execution phase will not ask such queries to unknown ciphertexts with high probability and we maintain the approximate correctness of the obfuscation.

In order to prove this upper bound on the number of ciphertexts that will be hit, we start with the query-adaptive A which consists of the *combination* of the learning and nal execution phases that accepts as input an obfuscation B in the oracle model and is able to adaptively query when running B on multiple randomly chosen inputs. We then show through a sequence of reductions to other adversaries that the advantage of such an attacker in hitting a speci c number of unknown ciphertexts is upper bounded by the advantage of a di erent non-adaptive attacker $\widehat{A}$ in hitting the same number of ciphertexts (up to some factor). We then nally show that $\widehat{A}$ has a negligible advantage in succeeding.

We begin by de ning the notion of query adaptivity for oracle algorithms and specify what it means for an adversary to hit a ciphertext.

Definition 8.4.19 (Query Adaptivity)**.** Let A be a poly-query randomized oracle algorithm that asks τ queries to some idealized oracle I. Suppose Q is the set of queries that A will ask. We de ne the level of *query adaptivity* of A as being one of two possible levels:

Non-adaptive: Q consists of τ queries, possibly from di erent domains, and chosen by A before it issues any query and/or independently of the answers of any previous query.

Fully adaptive: Q = $(q_1, ..., q$) consists of τ queries possibly from di erent domains where, for each i 2 $[\tau]$, q_{i+1} is determined by the answer returned by q_i.

Definition 8.4.20 (Ciphertext Hit)**.** Let A be a τ-query oracle algorithm that has access to . We say that A has *hit* a ciphertext c if it queries Dec$(., c)$, Rev(c), or RevMsg$(., ., c)$ and c is a valid unknown ciphertext (that is, A has never asked Enc(x) = c). We denote the set of ciphertexts that A has hit by H_A.

Our goal is to prove the following lemma which provides the desired upper bound on the number of ciphertexts that an attacker A can hit.

Lemma 8.4.21 (Hitting Ciphertexts)**.** *Let* $_{V,F}$ *be as in De nition 8.4.3,* n *be a xed number, and* $t(n)$ $p(n)$ $\omega(n)$*, where* t *is the upper bound on the output length of* F *and* p *is the ciphertext length. Let* A *be an adaptive* τ*-query oracle algorithm that takes as input* z *and has access to* $_{V,F}$*. Let* H_A *be the set of unknown valid ciphertexts that* A *hits. Then for security parameter (of the obfuscation scheme)* κ*,* n lg κ*,* τ poly(κ) $\kappa^{O(1)}$ *we have that for any* s τ*:*

$$\Pr[|H_A| \quad s] \quad O(2^{\quad (t+\ell(n))s})$$

where α = $|z|$ + $(t + 2n)s$*.*

Proof. We will de ne a sequence of adversaries and show reductions between them in order to prove the upper bound stated above. Throughout, we assume that the algorithms are in canonical form (see De nition 8.4.4).

1. **Attacker** A: This is the original adaptive τ-query attacker as de ned in the statement of the lemma where it will receive some input z and can ask τ queries to . The goal of the adversary is to hit at least s unknown valid ciphertexts via queries to Dec, Rev or RevMsg.

2. **Attacker** A_u: This is the same attacker as A but does not accept any input and is modi ed as follows. For any Dec, Rev or RevMsg queries asked to with some answer y 6 ?, A_u will instead use an answer that is part of some xed string u 2 $\{0,1\}$ hardcoded within A_u where α = $|z|$ + $(t + 2n)s$. The Enc queries are handled normally as before. The goal of this adversary is to hit at least s unknown valid ciphertexts via queries to Dec, Rev or RevMsg.

3. **Attacker $A^{\emptyset}$**: This is the same attacker as A_u for any xed u. However, aside from Enc queries which are handled normally using , the other query types are instead replaced with a single subroutine Test that takes as input a ciphertext c and outputs 1 if c is valid, and 0 otherwise. The goal of this adversary is to hit at least s unknown valid ciphertexts via queries to Test.

4. **Attacker $\widehat{A}$** : This is the *non-adaptive* attacker where it will ask all its queries at once at the start of the experiment. Furthermore, it will *not* ask any Enc queries but will be constrained to asking only Test queries. The goal of this adversary is to hit at least s unknown valid ciphertexts via queries to Test.

Lemma 8.4.22. *For every A, there exists some u 2 f0,1g such that* Pr[$jH_A j$ s] 2 Pr[$jH_{A_u} j$ s]

Proof. Recall that A accepts z as input and, when it hits s ciphertexts, it would receive back at most (t + 2n) since we can either get back t bits information as a result of getting back an answer from $\mathsf{Dec}_{\mathsf{V;F}}$ or at most n bits of information from queries of Rev and $\mathsf{RevMsg}_{\mathsf{V}}$. Furthermore, by the canonicalization of A, it can ask for any c at most one query of each type $\mathsf{Dec}_{\mathsf{V;F}}$, Rev, and $\mathsf{RevMsg}_{\mathsf{V}}$. Thus, in order to say that A_u would succeed at hitting s with the same amount of information, the length of u has to be α = jzj + (t + 2n)s. Now, by a union bound over all u, the probability of success for A is given as follows:

$$\Pr[jH_A j \quad s] \quad \Pr[9\, u : jH_{A_u} j \quad s] \quad \sum_u \Pr[jH_{A_u} j \quad s] \quad 2 \ \Pr[jH_{A_u} j \quad s]$$

□

Lemma 8.4.23. *For any u 2 f0,1g ,* Pr[$jH_{A_u} j$ s] = Pr[$jH_{A^{\emptyset}} j$ s]

Proof. Since A_u does not obtain any information regarding the actual answers to the Dec, Rev and RevMsg queries that it asks, we can think of these subroutines simply as a testing procedure that A_u can use to determine whether any given ciphertext c is valid or not, and this is signaled by whether the oracle returns *?* or not to any of these queries. Therefore, we can interpret A_u as an adversary $A^{\emptyset}$ that simply calls Test instead of Dec, Rev and RevMsg queries as this yields the same result. □

Lemma 8.4.24. Pr[$jH_{A^{\emptyset}} j$ s] Pr[$\left|H_{\widehat{A}}\right|$ s]

Proof. Given attacker $A^{\emptyset}$ we can de ne $\widehat{A}$ that uses $A^{\emptyset}$ and only issues Test queries (non-adaptively). Any Enc queries that $A^{\emptyset}$ asks (from a speci c Enc domain of size n) can be lazily evaluated (emulated) by $\widehat{A}$. Furthermore, any Test queries that $A^{\emptyset}$ asks will be answered using one of $\widehat{A}$'s pre-issued Test queries while remaining consistent with the previous Enc queries that were issued. □

Lastly, we state and prove the following lemma which will be used to bound the number of ciphertexts that any (poly-query) non-adaptive algorithm might obtain and use for its decryption and/or reveal queries.

Lemma 8.4.25 (Hitting Ciphertexts for Non-Adaptive Learners). *Let be as in De nition 8.4.2 and* $t(n)$ $p(n)$ $\omega(n)$ *where* t *is an upper bound on the output length of* F *and* p *is the ciphertext length. Let* $\widehat{A}$ *be a non-adaptive* τ*-query canonical algorithm as de ned above and* $H_{\widehat{A}}$ *be the set of unknown valid ciphertexts that* $\widehat{A}$ *hits via* Test *queries. Then for security parameter* κ*, xed* n lg κ*,* τ poly(κ)*, we have that for any* s τ*:*

$$\Pr[\left|H_{\widehat{A}}\right| \quad s] \quad O(2^{\ (t+\,!\,(n))s})$$

Proof. Suppose t p dn for $d = \omega(1)$ and let τ $\kappa^{d'} = 2^{d' \lg}$ $2^{d'n}$ where $d' = d/2 = \omega(1)$ for the purposes of upper-bounding the probability for all poly-query algorithms $\widehat{A}$. Recall that the function Enc(.) is injective and maps messages x 2 $f0,1g^n$ to ciphertexts c 2 $f0,1g^{p(n)}$. For simplicity, assume that we want to compute the probability that $jH_{\widehat{A}}j = s$. For any set of s ciphertexts that are in the image of some xed s-sized set of the domain Enc(.), the probability that the τ queries will hit these s ciphertexts is given by $\binom{}{s}/\binom{2^p}{s}$. By a union bound over all the di erent s-sized sub-domains of Enc(.), we nd that for su ciently large security parameter κ:

$$\Pr[\left|H_{\widehat{A}}\right| = s] \quad \binom{2^n}{s}\frac{\binom{}{s}}{\binom{2^p}{s}} \quad \frac{\left(\frac{2^n e}{s}\right)^s\left(\frac{\tau e}{s}\right)^s}{\left(\frac{2^p}{s}\right)^s} \quad \left(\frac{\frac{2^n e}{s}\quad\frac{2^{d'}n e}{s}}{\frac{2^p}{s}}\right)^s \quad \left(\frac{2^{n(1+d')}e^2}{2^p s}\right)^s$$

$$\left(\frac{2^{n(1+d=2)}e^2}{2^p}\right)^s \quad O(2^{\ (t+\,!\,(n))s})$$

The last inequality follows from the short-output property, that is t p d n for some $d = \omega(1)$. Note that $\Pr[jH_{\widehat{A}}j = s+1]$ $\Pr[jH_{\widehat{A}}j = s]$ and therefore $\Pr[jH_{\widehat{A}}j$ $s]$ is dominated by the largest term represented by $\Pr[jH_{\widehat{A}}j = s]$. □

Putting things together. By Lemmas 8.4.22, 8.4.23, and 8.4.24, and using Lemma 8.4.25, we nd that:

$$\Pr[jH_Aj \quad s] \quad O(2^{\ (t+\,!\,(n))s})$$

Note that, for simplicity, Lemma 8.4.21 only considers hitting unknown ciphertexts from some xed domain of size n. However, we observe that this argument can be extended for learners that can ask queries for di erent domain sizes as well.

□

Lemma 8.4.26. $\Pr[E(x)]$ ϵ + negl(κ)

Proof. Let A to be an adaptive non-uniform oracle algorithm in the ideal hybrid that has access to and works as follows:

> Initialize the query-answer set Q_A = ?
> For i = $f1,...,kg$, run Ev (B,z_i). For any query q asked by Ev (B,z_i), if $(q$ 7! $a)_T$ 2 Q_A for subroutine T then answer with a. Otherwise, handle the query in the canonical form as in De nition 8.4.4, and if a query was sent to , add the new query-answer pair $(q$ 7! $a)_T$ to Q_A.

Output Ev (B, z_k)

In essence, A would run the learning and nal execution phases (in total k executions) making sure to only forward to the queries that are distinct and which cannot be computed from Q_A so far. Given the above canonical A, we observe that for any unknown valid ciphertext c = Enc(x) where x = (a, m), A would ask at most one query of the form Rev(c), at most one query of the form Dec(w, c) for which $V^{Enc}(w, a)$ = 1, and at most one query of the form RevMsg(w_1, w_2, c) for which $V^{Enc}(w_i, a)$ = 1 where i 2 f1, 2g. Furthermore, A would never ask a query if $V^{Enc}(w, a)$ = 0 since this condition can be veri ed independently by A and the answer can be simulated as it would invariably be ?.

Given A, we can bound the number of distinct unknown ciphertexts that the k executions will hit, which we denote by $jH_Bj = \left|\bigcup_{i=1}^{k} H_{B_i}\right|$ where H_{B_i} is the set of ciphertexts hit by the ith evaluation Ev (B, z_i). Note that the total number of queries that will be asked across all executions is $k\ell_B$ = poly(κ) where ℓ_B is the circuit size of Ev$(B, .)$. It is straightforward to see that, for any s, Pr[jH_Aj s] = Pr[jH_Bj s] since whenever one of the k executions hits an unknown ciphertext c for this rst time, A will also forward it to the oracle and hit it for the rst time as well.

Since A accepts as input the obfuscated circuit of size $jiOj = \ell_O$, by Lemma 8.4.21, the probability that A hits at least s = (ℓ_O + κ) ciphertexts is at most $2^{\ell_O - !(n)s}$ $2^{-!(n)}$ = negl(κ). Therefore, the $k\ell_B$-query algorithm A will hit at most s = (ℓ_O + κ) new unknown ciphertexts with overwhelming probability. Therefore we have that,

$$\Pr[jH_Bj \quad s] = \Pr[jH_Aj \quad s] \quad 2^{\ell_O - !(n)s}$$

Since the maximum possible number of learning iterations k = $(\ell_O+\kappa)/\epsilon > s$ and $\bigcup_{j=1}^{i} H_{B_j}$ $\bigcup_{j=1}^{i+1} H_{B_j}$ for any i, the number of learning iterations that increase the size of the set H_B of unknown ciphertext hits (via one of the bad event queries) is at most s. A ciphertext that was hit could have its encryption query generated during the obfuscation phase or as one of the hidden queries issued by F during one of the k executions. We say $\lambda \xleftarrow{\$} [k]$ is bad if it is the case that $\bigcup_{j=1} H_{B_j}$ $\bigcup_{j=1}^{+1} H_{B_j}$ (i.e. λ is an index of a learning iteration that increases the size of the hit ciphertexts). This would imply that after λ learning iterations in the ideal experiment, the nal execution with $H_{\widehat{B}} := \bigcup_{j=1}^{+1} H_{B_j}$ would contain an unknown ciphertext that it we will hit for this rst time and for which we cannot consistently answer the queries that reference it. Thus, given that we have set k = (ℓ_O + κ)/ϵ, the probability (over the selection of λ) that λ is bad is at most $s/k < \epsilon$.

□

Proving Security. To show that the resulting obfuscator is secure, it su ces to show that the compilation process represented as the new obfuscator's construction is simulatable. We show a simulator Sim (with access to) that works as follows: given an obfuscated circuit B in the ideal model, it runs the learning procedure as shown in Step 2 of the new obfuscator $\widehat{iO}$ to learn the heavy queries Q_B then outputs $\widehat{B}$ = (B, Q_B). Note that this distribution is statistically close to the output of the real execution of $\widehat{iO}$ and, therefore, security follows.

8.5 Extended Long-Output FE Implies Obfuscation

We show that for a certain choice of function family G such that outlen(g) m + 1 for any g 2 G (where outlen(.) denotes the output length of a given function), there exists a monolithic construction of IO from a compact single-key FE with ciphertext length m for the function family G.

Theorem 8.5.1. *There exists a function family* G *with* outlen(g) m + 1 *for any* g 2 G *such that an* m *ciphertext length compact single-key FE for the function family* G *implies IO in a monolithic way.*

In Lemma 8.5.2, we show that extended FE of the kind speci ed in the theorem above implies extended compact single-key FE. This, together with the fact that extended compact FE implies IO [AJ15, BV15], allows us to conclude the above theorem.

Lemma 8.5.2. *For any function family* H *with* outlen(h) ℓ *for any* h 2 H*, there exists a class of functions* $f g_i g_{i2[\ell]}$ *with* outlen(g_i) m + 1 *for all* i 2 [ℓ] *such that there exists a monolithic construction of extended compact single-key FE for the function family* H *from any* m*-ciphertext length compact single-key FE for the class of functions* $f g_i g_{i2[\ell]}$.

Given an extended FE (Setup, Enc, Dec) of the kind speci ed in the above lemma statement, and a secret-key encryption scheme (S, E, D)[4] we construct an extended compact single-key FE scheme ($\mathsf{Setup}_H, \mathsf{Enc}_H, \mathsf{Dec}_H$). In a bit more detail, realizing extended compact single-key FE for the family H (where each h 2 H could have $\mathsf{Setup}_H, \mathsf{Enc}_H$, and Dec_H gates in it) with output length ℓ requires ℓ instances of the given FE scheme corresponding to functions $f g_i g_{i2[\ell]}$. We describe these functions in Figure 8.2.

Construction of Compact FE. Next, we describe the construction of our compact single-key FE scheme ($\mathsf{Setup}_H, \mathsf{Enc}_H, \mathsf{Dec}_H$):

$\mathsf{Setup}_H(1\ , h)$: Sample s $S(1\ )$, set $\mathsf{PK}_{\ell+1}$ = ?, and for each i = $f\ell \ldots 1g$ proceed as follows:

1. $(\mathsf{PK}_i, \mathsf{SK}_i)$ Setup$(1\ , g_i[i, h, \alpha_i, \mathsf{PK}_{i+1}])$, where α_i $E(s, 0^{m+1})$.

Output $f(\mathsf{PK}_H = \mathsf{PK}_1, \mathsf{SK}_H = (\mathsf{SK}_1, \ldots, \mathsf{SK}_\ell))g$

$\mathsf{Enc}_H(\mathsf{PK}_H = \mathsf{PK}_1, y)$: Given the public key PK_H and input x, sample r $f0, 1g$ and output ciphertext ct = $\mathsf{Enc}(\mathsf{PK}_1, (0, y, r, 0\ ))$.

$\mathsf{Dec}_H(\mathsf{SK}_H = (\mathsf{SK}_1, \ldots, \mathsf{SK}_\ell), ct)$: Let ct_1 = ct and then for each i = $f1 \ldots \ell g$ proceed as follows:

[4] $S(1^\kappa)$ on input the security parameter outputs a secret key s. $E(s, m)$ uses the secret-key s to encrypt m generating a ciphertext c. Finally, the generated ciphertext c can be decrypted using $D(s, c)$ to recover the original message m back. Note that such an encryption scheme is implied by a underlying single-key FE scheme.

$g_i[i, h, \alpha, \mathsf{PK}](b, y, r, s)$

Hardcoded Parameters: The index i, function h 2 H, a ciphertext string α, and a public-key PK of the underlying single-key FE scheme.

Input: A bit b 2 f0, 1g, input y in the domain of h, randomness r 2 f0, 1g and a secret key s of the secret-key encryption.

Computation: It proceeds as follows:

1. If b = 0 and $i < \ell$, output a_ikEnc(PK, (0, y, r, 0); PRF$_r$(i)), where a = h(y) and a_i is its i^{th} bit.
2. If b = 0 and i = ℓ, output a_ℓk0^m where a = h(y) and a_ℓ is its ℓ^{th} bit.
3. Else, output $D(s, \alpha)$.

Figure 8.2: The output-constrained circuit g_i de ned for the underlying compact FE scheme.

1. Parse Dec(SK$_i$, ct_i) as a_ikct_{i+1}.

Output a_1 a_ℓ.

Gates planted in h. Note that, since we are constructing an *extended* compact FE, h can have Setup$_\mathsf{H}$, Enc$_\mathsf{H}$, and Dec$_\mathsf{H}$ gates in it. Given the description of our scheme above we can replace each of these gates by Setup, Enc, and Dec gates | namely, the gates of the underlying FE scheme. This allows us to conclude that each g_i consists of only Setup, Enc, Dec, E, and D gates. However, since (S, E, D) is obtained by using Setup, Enc, and Dec, we have that the set of gates reduces to Setup, Enc, and Dec. Note that since we are assuming that the underlying FE scheme is extended this is allowed.

Efficiency. Observe that a ciphertext of the constructed FE scheme (Setup$_\mathsf{H}$, Enc$_\mathsf{H}$, Dec$_\mathsf{H}$) for the circuit h consists only of a single ciphertext of the underlying FE scheme (Setup, Enc, Dec) for the circuit g_1. Therefore, we can conclude that the size of the encryption circuit of the constructed scheme is independent of the size of the circuit h.

Correctness. The correctness of our scheme follows directly from the correctness of the underlying FE scheme. Observe that for each i decryption of ciphertext ct_i generates the i^{th} bit of the output of h on input y and the correctly formed ciphertext ct_{i+1}. This, by induction, allows us to conclude that decryption yields the correct output.

Proof of Security. We need to prove that the distributions f(PK$_\mathsf{H}$, SK$_\mathsf{H}$, Enc$_\mathsf{H}$(PK$_\mathsf{H}$, y_0)g and f(PK$_\mathsf{H}$, SK$_\mathsf{H}$, Enc$_\mathsf{H}$(PK$_\mathsf{H}$, y_1)g are computationally indistinguishable. We prove security via a sequence of hybrids.

H_0: This hybrid corresponds to the distribution $f\mathsf{PK}_H, \mathsf{SK}_H, ct_1 = \mathsf{Enc}_H(\mathsf{PK}_H, y_0)g$, where $\mathsf{PK}_H = \mathsf{PK}_1$ and $\mathsf{SK}_H = (\mathsf{SK}_1, \ldots, \mathsf{SK}_\ell)$. Also, recall that $\mathsf{PK}_i, \mathsf{SK}_i$ for each $i \in [\ell]$ is an instance of the underlying FE scheme for the circuit $g_i[i, h, \alpha_i, \mathsf{PK}_{i+1}]$. Finally, let $a_i \| ct_{i+1}$ for $i \in f1 \ldots \ell g$ be the output of $\mathsf{Dec}(\mathsf{SK}_i, ct_i)$.

H_1: In this hybrid, we change α_i for each $i \in [\ell]$ (the value hard-coded in the circuit g_i). Recall that in hybrid H_0, α_i is obtained as $E(s, 0^{m+1})$. Now we instead generate α_i as $E(s, a_i \| ct_{i+1})$, where $a_i \| ct_{i+1}$ is the output from decryption of ct_i with secret-key SK_i.

Indistinguishability between hybrids H_0 and H_1 follows based on the semantic security of the secret-key encryption scheme (S, E, D).

$H_{2,i}$: For $i \in f1 \ldots \ell g$ this hybrid is same as hybrid $H_{2,i-1}$ except that we change how ct_i is generated. Specifically, we generate $ct_i = \mathsf{Enc}(\mathsf{PK}_i, (1, 0^n, r, s))$ instead of $\mathsf{Enc}(\mathsf{PK}_i, (0, y_0, r, 0\ ); \mathsf{PRF}_r(i - 1))$ when $i \neq 1$ and $\mathsf{Enc}(\mathsf{PK}_i, (0, y_0, r, 0\ ))$ when $i = 1$.[5] Note that this is a sequence of $\ell + 1$ hybrids where $H_{2,0}$ is same as hybrid H_1.

Next we show that hybrid $H_{2,i}$ is indistinguishable from hybrid $H_{2,i+1}$ for each $i \in f0 \ldots \ell g$ relying on the security of the PRF and the underlying single-key FE. More specifically, observe that an attacker distinguishing $H_{2,i}$ and $H_{2,i+1}$ can be used to distinguish the following two distributions: $(\mathsf{PK}_i, \mathsf{SK}_i, \mathsf{Enc}(\mathsf{PK}_i, (0, y_0, r, 0\ ); \mathsf{PRF}_r(i - 1))$ (or $(\mathsf{PK}_i, \mathsf{SK}_i, \mathsf{Enc}(\mathsf{PK}_i, (0, y_0, r, 0\ ))$ when $i = 1$) and $(\mathsf{PK}_i, \mathsf{SK}_i, \mathsf{Enc}(\mathsf{PK}_i, (1, 0^n, r, s))$, while we have that $g_i[i, h, \alpha_i, \mathsf{PK}_{i+1}](0, y_0, r, 0\ ) = g_i[i, h, \alpha_i, \mathsf{PK}_{i+1}](1, 0^n, r, s)$.

Now consider the intermediate hybrid $H^0_{2,i}$ with the distribution given by the tuple $(\mathsf{PK}_i, \mathsf{SK}_i, \mathsf{Enc}(\mathsf{PK}_i, (0, y_0, r, 0\ ))$. It is easy to see that an attacker distinguishing $H_{2,i}$ and $H^0_{2,i}$ for $i \in f1 \ldots \ell g$ can be used to break the security of the PRF.

Next, since $g_i[i, h, \alpha_i, \mathsf{PK}_{i+1}](0, y_0, r, 0\ ) = g_i[i, h, \alpha_i, \mathsf{PK}_{i+1}](1, 0^n, r, s)$, therefore we can conclude that an attacker distinguishing between $H^0_{2,i}$ and $H_{2,i+1}$ can be used to break the indistinguishability security of the underlying FE scheme.

Note that hybrid $H_{2,\ell}$ is independent of y_0. In other words, starting hybrid H_0 with input y_1 and making the same changes as above still yields hybrid $H_{2,\ell}$. This allows us to conclude that distribution from hybrid H_0, namely $f\mathsf{PK}_H, \mathsf{SK}_H, ct_1 = \mathsf{Enc}_H(\mathsf{PK}_H, y_0)g$, is computationally indistinguishable from the distribution $f\mathsf{PK}_H, \mathsf{SK}_H, ct_1 = \mathsf{Enc}_H(\mathsf{PK}_H, y_1)g$.

8.6 Fully Black-Box Separation of IO from Functional Encryption

In contrast to Section 8.4 where we show that a monolithic construction of IO from short-output functional encryption is impossible, in this section we show that a fully black-box

[5] Here n is the length of the input x.

construction of IO from (not necessarily short output) functional encryption is impossible. More formally, we prove the following theorem.

Theorem 8.6.1. *Assume the existence of one-way functions and that* $\mathbf{NP} \not\subseteq \mathbf{coAM}$. *Then there exists no fully black-box construction of indistinguishability obfuscation (IO) from functional encryption.*

To prove Theorem 8.6.1, we will apply Lemma 7.2.7 for the idealized random (non-extended) functional witness encryption (rFWE) model $\bar{}$ (see De nition 8.4.2) to prove that there is no black-box construction of IO from any primitive P that can be constructed from $\bar{}$ (with measure one over the oracle). In particular, we will do so for P that is the functional encryption primitive. Our task is thus twofold: (1) to prove that (non-extended) functional encryption can be constructed from $\bar{}$ and (2) to show a simulatable compilation procedure that compiles out $\bar{}$ from any IO construction. By Lemma 7.2.7, this would imply the separation result of IO from P and prove Theorem 8.6.1.

8.6.1 Single-Key (Non-Extended) Functional Encryption exists relative to $\bar{}$

In Section 8.4.2, we have shown that extended functional encryption is secure relative to an extended version of . This was done in two steps: we rst proved that a secure construction of customized functional witness encryption relative to , then we proved that there exists a fully black-box construction of an extended functional encryption scheme from any customized functional witness encryption scheme.

We observe that the proof applied there also applies here in a direct sense, speci cally because the (non-extended) functional encryption scheme only requires a non-extended functional witness encryption scheme (and one-way functions). Hence, for the rst step, when we want to construct the (non-extended) FWE scheme using $\bar{}$, we can use exactly Construction 8.4.10 and the proofs of correctness and security would follow directly from Claim 8.4.11 as this is a special case of the monolithic construction.

Furthermore, for the second step, we can also use Construction 8.4.15 to construct our (non-extended) functional encryption scheme but with the added restriction that F, the functional family supported by the single-key FE scheme, is not allowed make oracle calls to $\bar{}$. The proofs of correctness and security would follow directly from Claim 8.4.16 as this is a special case of the monolithic construction.

8.6.2 Compiling out $\bar{}$ from IO

Similar to how we compiled out the extended oracle in Section 8.4.5, we will apply the same techniques here for compiling out $\bar{}$ in two steps: we rst compile out some of the subroutines of $\bar{}$ to get an approximate obfuscation in the random instance-revealing witness encryption oracle model then compile out to get an approximate obfuscator in the plain model. More formally, we prove the following claim.

Claim 8.6.2. *Let be a random instance-revealing witness encryption oracle. Then for any non-extended* $\bar{}_{V;F;p}$ = (Enc, $\text{Dec}_{V;F}$, Rev, RevMsg_V) *and , the following holds:*

> *For any IO in the non-extended* $\bar{}_{V;F;p}$ *ideal model, there exists a simulatable compiler with correctness error* $\epsilon < 1/200$ *for it that outputs a new obfuscator in the oracle model.*
>
> *For any IO in the oracle model, there exists a simulatable compiler with correctness error* $\epsilon < 1/200$ *for it that outputs a new obfuscator in the plain model.*

Proof. As before, we will only prove the rst part of the claim as the second part follows from Lemma 7.3.13. However, since this is a signi cantly simpli ed version of Claim 8.4.17, we will only give a high-level overview the compilation procedure and the proof of approximate correctness. The compiler will behave exactly as described in Section 8.4.5 but now we only consider V that has Enc gates and F that does not make any oracle calls (only accepts plain circuits).

Recall that the emulation process that outputs B iO$\bar{}$(C) and the evaluations during the learning phase Ev$\bar{}$(B, z_1), ..., Ev$\bar{}$(B, z), and nal execution phase are all canonicalized as in De nition 8.4.4. Hence, since there are no queries that can be hidden during the learning and nal execution phases (due to the canonicalization and the non-extended nature of $\bar{}$) the only error in correctness of the nal execution will be due to asking Dec on an unknown ciphertext c where c was generated during the emulation of iO$\bar{}$(C). Note that asking a valid unknown ciphertext c that was never generated happens with negligible probability due to the injectivity of Enc.

If we consider the combination of all the learning and nal execution phases as one algorithm then, by Remark 8.4.5, for any unknown valid ciphertext c, there can be at most one query of the form Dec($., c$). Since we can upper bound the number of unknown ciphertexts by ℓ_O = *j*iO*j* to be the size of the obfuscator, we can apply the learning procedure for $k = 3\ell_O/\epsilon$ to get an ϵ error in the correctness of the new obfuscator. □

Chapter 9

Conclusion

In this dissertation we have made signi cant progress towards understanding the computational complexity of indistinguishability obfuscation (IO), which is an enabler of many sought-after powerful applications in cryptography. However, as useful as this object is, our work has provided an explanation behind the lack of candidate constructions of IO that are based on standard cryptographic assumptions (in fact, all currently known candidate constructions of IO are based on strong, unconventional assumptions of questionable security). In particular we have shown that standard classical assumptions, such as one-way functions and trapdoor permutations, cannot be used in a black-box way to get IO. Furthermore, we go even further and show that modern standard assumptions such as homomorphic encryption and predicate encryption are insu cient to obtain IO when used in a monolithic way that captures a large and natural class of non-black-box techniques. While our results lends evidence that IO may in fact require unorthodox tools and assumptions to realize, there still exists the possibility that it may be based on well-studied lattice-based assumptions such as the learning with errors problem, and we leave this as a highly interesting open question.

Furthermore, we strongly believe that the monolithic model, which was developed in this work as a means of proving separation results of IO from modern assumptions, will be a useful tool in proving separation results for other primitives that use their underlying components in some non-black-box way. In fact, one can also go as far as to extend existing classical impossibility results (e.g. black-box impossibility of public-key encryption from one-way functions) to possibly prove that a large class of non-black-box techniques are insu cient for achieving the desired primitive.

Bibliography

[AB15] Benny Applebaum and Zvika Brakerski. Obfuscating circuits via composite-order graded encoding. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015: 12th Theory of Cryptography Conference, Part II*, volume 9015 of *Lecture Notes in Computer Science*, pages 528{556, Warsaw, Poland, March 23{25, 2015. Springer, Heidelberg, Germany. 2

[ABG+13] Prabhanjan Ananth, Dan Boneh, Sanjam Garg, Amit Sahai, and Mark Zhandry. Di ering-inputs obfuscation and applications. Cryptology ePrint Archive, Report 2013/689, 2013. http://eprint.iacr.org/2013/689. 95

[AGIS14] Prabhanjan Vijendra Ananth, Divya Gupta, Yuval Ishai, and Amit Sahai. Optimizing obfuscation: Avoiding Barrington's theorem. In Gail-Joon Ahn, Moti Yung, and Ninghui Li, editors, *ACM CCS 14: 21st Conference on Computer and Communications Security*, pages 646{658, Scottsdale, AZ, USA, November 3{7, 2014. ACM Press. 2

[AJ15] Prabhanjan Ananth and Abhishek Jain. Indistinguishability obfuscation from compact functional encryption. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *Advances in Cryptology { CRYPTO 2015, Part I*, volume 9215 of *Lecture Notes in Computer Science*, pages 308{326, Santa Barbara, CA, USA, August 16{20, 2015. Springer, Heidelberg, Germany. 2, 77, 79, 83, 87, 152, 153, 154, 178

[AJS15] Prabhanjan Ananth, Abhishek Jain, and Amit Sahai. Indistinguishability obfuscation from functional encryption for simple functions. Cryptology ePrint Archive, Report 2015/730, 2015. http://eprint.iacr.org/2015/730. 83

[AJS17] Prabhanjan Ananth, Abhishek Jain, and Amit Sahai. Indistinguishability obfuscation for turing machines: Constant overhead and amortization. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology { CRYPTO 2017, Part II*, volume 10402 of *Lecture Notes in Computer Science*, pages 252{279, Santa Barbara, CA, USA, August 20{24, 2017. Springer, Heidelberg, Germany. 77, 79

[Ale03] Michelle Alexopoulos. Notes on set theory and probability theory, 2003. http://www.biostat.umn.edu/~dipankar/pubh7440/ProbSets.pdf. 11

[App14] Benny Applebaum. Bootstrapping obfuscators via fast pseudorandom functions. In Palash Sarkar and Tetsu Iwata, editors, *Advances in Cryptology { ASIACRYPT 2014, Part II*, volume 8874 of *Lecture Notes in Computer Science*, pages 162{172, Kaoshiung, Taiwan, R.O.C., December 7{11, 2014. Springer, Heidelberg, Germany. 79

[AS15] Gilad Asharov and Gil Segev. Limits on the power of indistinguishability obfuscation and functional encryption. In Venkatesan Guruswami, editor, *56th Annual Symposium on Foundations of Computer Science*, pages 191{209, Berkeley, CA, USA, October 17{20, 2015. IEEE Computer Society Press. 5, 8, 78, 83

[AS16] Gilad Asharov and Gil Segev. On constructing one-way permutations from indistinguishability obfuscation. In Eyal Kushilevitz and Tal Malkin, editors, *TCC 2016-A: 13th Theory of Cryptography Conference, Part II*, volume 9563 of *Lecture Notes in Computer Science*, pages 512{541, Tel Aviv, Israel, January 10{13, 2016. Springer, Heidelberg, Germany. 78, 83

[AS17] Prabhanjan Ananth and Amit Sahai. Projective arithmetic functional encryption and indistinguishability obfuscation from degree-5 multilinear maps. In Jean-Sebastien Coron and Jesper Buus Nielsen, editors, *Advances in Cryptology { EUROCRYPT 2017, Part I*, volume 10210 of *Lecture Notes in Computer Science*, pages 152{181, Paris, France, May 8{12, 2017. Springer, Heidelberg, Germany. 153, 154

[AW07] Ben Adida and Douglas Wikstrom. How to shu e in public. In Salil P. Vadhan, editor, *TCC 2007: 4th Theory of Cryptography Conference*, volume 4392 of *Lecture Notes in Computer Science*, pages 555{574, Amsterdam, The Netherlands, February 21{24, 2007. Springer, Heidelberg, Germany. 31

[BBF13] Paul Baecher, Christina Brzuska, and Marc Fischlin. Notions of black-box reductions, revisited. In Kazue Sako and Palash Sarkar, editors, *Advances in Cryptology { ASIACRYPT 2013, Part I*, volume 8269 of *Lecture Notes in Computer Science*, pages 296{315, Bengalore, India, December 1{5, 2013. Springer, Heidelberg, Germany. 26

[BBF16] Zvika Brakerski, Christina Brzuska, and Nils Fleischhacker. On statistically secure obfuscation with approximate correctness. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology { CRYPTO 2016, Part II*, volume 9815 of *Lecture Notes in Computer Science*, pages 551{578, Santa Barbara, CA, USA, August 14{18, 2016. Springer, Heidelberg, Germany. 6, 8, 15, 63, 64, 65, 72, 73, 74, 96, 97, 98, 156

[BC10] Nir Bitansky and Ran Canetti. On strong simulation and composable point obfuscation. In Tal Rabin, editor, *Advances in Cryptology { CRYPTO 2010*,

volume 6223 of *Lecture Notes in Computer Science*, pages 520{537, Santa Barbara, CA, USA, August 15{19, 2010. Springer, Heidelberg, Germany. 1, 31

[BCP14] Elette Boyle, Kai-Min Chung, and Rafael Pass. On extractability obfuscation. In Yehuda Lindell, editor, *TCC 2014: 11th Theory of Cryptography Conference*, volume 8349 of *Lecture Notes in Computer Science*, pages 52{73, San Diego, CA, USA, February 24{26, 2014. Springer, Heidelberg, Germany. 22, 95, 155

[BF01] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil pairing. In Joe Kilian, editor, *Advances in Cryptology { CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 213{229, Santa Barbara, CA, USA, August 19{23, 2001. Springer, Heidelberg, Germany. 62

[BGI+01] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In Joe Kilian, editor, *Advances in Cryptology { CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 1{18, Santa Barbara, CA, USA, August 19{23, 2001. Springer, Heidelberg, Germany. 1, 6, 14, 31

[BGK+14] Boaz Barak, Sanjam Garg, Yael Tauman Kalai, Omer Paneth, and Amit Sahai. Protecting obfuscation against algebraic attacks. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology { EUROCRYPT 2014*, volume 8441 of *Lecture Notes in Computer Science*, pages 221{238, Copenhagen, Denmark, May 11{15, 2014. Springer, Heidelberg, Germany. 4, 6, 32, 49

[BK95] Manuel Blum and Sampath Kannan. Designing programs that check their work. *J. ACM*, 42(1):269{291, January 1995. 62

[BKSY11] Zvika Brakerski, Jonathan Katz, Gil Segev, and Arkady Yerukhimovich. Limits on the power of zero-knowledge proofs in cryptographic constructions. In Yuval Ishai, editor, *TCC 2011: 8th Theory of Cryptography Conference*, volume 6597 of *Lecture Notes in Computer Science*, pages 559{578, Providence, RI, USA, March 28{30, 2011. Springer, Heidelberg, Germany. 5, 8, 78, 83

[BLP17] Nir Bitansky, Huijia Lin, and Omer Paneth. On removing graded encodings from functional encryption. In Jean-Sebastien Coron and Jesper Buus Nielsen, editors, *Advances in Cryptology { EUROCRYPT 2017, Part II*, volume 10211 of *Lecture Notes in Computer Science*, pages 3{29, Paris, France, May 8{12, 2017. Springer, Heidelberg, Germany. 152

[BMG07] Boaz Barak and Mohammad Mahmoody-Ghidary. Lower bounds on signatures from symmetric primitives. In *48th Annual Symposium on Foundations of Computer Science*, pages 680{688, Providence, RI, USA, October 20{23, 2007. IEEE Computer Society Press. 67

[BMSZ15] Saikrishna Badrinarayanan, Eric Miles, Amit Sahai, and Mark Zhandry. Post-zeroizing obfuscation: The case of evasive circuits. Cryptology ePrint Archive, Report 2015/167, 2015. http://eprint.iacr.org/2015/167. 2

[BNPW16] Nir Bitansky, Ryo Nishimaki, Alain Passelegue, and Daniel Wichs. From cryptomania to obfustopia through secret-key functional encryption. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B: 14th Theory of Cryptography Conference, Part II*, volume 9986 of *Lecture Notes in Computer Science*, pages 391{418, Beijing, China, October 31 { November 3, 2016. Springer, Heidelberg, Germany. 152

[Bor09] Emile Borel. Les probabilites denombrables et leurs applications arithmetiques. *Rendiconti del Circolo Matematico di Palermo (1884-1940)*, 27(1):247{271, 1909. 11

[BP13] Nir Bitansky and Omer Paneth. On the impossibility of approximate obfuscation and applications to resettable cryptography. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th Annual ACM Symposium on Theory of Computing*, pages 241{250, Palo Alto, CA, USA, June 1{4, 2013. ACM Press. 4, 32, 33, 43, 53

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing e cient protocols. In V. Ashby, editor, *ACM CCS 93: 1st Conference on Computer and Communications Security*, pages 62{73, Fairfax, Virginia, USA, November 3{5, 1993. ACM Press. 28, 61

[BR13] Zvika Brakerski and Guy N. Rothblum. Obfuscating conjunctions. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology { CRYPTO 2013, Part II*, volume 8043 of *Lecture Notes in Computer Science*, pages 416{434, Santa Barbara, CA, USA, August 18{22, 2013. Springer, Heidelberg, Germany. 1

[BR14] Zvika Brakerski and Guy N. Rothblum. Virtual black-box obfuscation for all circuits via generic graded encoding. In Yehuda Lindell, editor, *TCC 2014: 11th Theory of Cryptography Conference*, volume 8349 of *Lecture Notes in Computer Science*, pages 1{25, San Diego, CA, USA, February 24{26, 2014. Springer, Heidelberg, Germany. 4, 6, 32, 49

[BSW07] John Bethencourt, Amit Sahai, and Brent Waters. Ciphertext-policy attribute-based encryption. In *2007 IEEE Symposium on Security and Privacy*, pages 321{334, Oakland, CA, USA, May 20{23, 2007. IEEE Computer Society Press. 24, 84

[BSW11] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: De nitions and challenges. In Yuval Ishai, editor, *TCC 2011: 8th Theory of Cryptography*

Conference, volume 6597 of *Lecture Notes in Computer Science*, pages 253{273, Providence, RI, USA, March 28{30, 2011. Springer, Heidelberg, Germany. 6, 17, 18

[BV15] Nir Bitansky and Vinod Vaikuntanathan. Indistinguishability obfuscation from functional encryption. In Venkatesan Guruswami, editor, *56th Annual Symposium on Foundations of Computer Science*, pages 171{190, Berkeley, CA, USA, October 17{20, 2015. IEEE Computer Society Press. 2, 21, 77, 79, 83, 87, 152, 153, 154, 178

[BY93] Mihir Bellare and Moti Yung. Certifying cryptographic tools: The case of trapdoor permutations. In Ernest F. Brickell, editor, *Advances in Cryptology { CRYPTO'92*, volume 740 of *Lecture Notes in Computer Science*, pages 442{460, Santa Barbara, CA, USA, August 16{20, 1993. Springer, Heidelberg, Germany. 61

[BZ14] Dan Boneh and Mark Zhandry. Multiparty key exchange, e cient traitor tracing, and more from indistinguishability obfuscation. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology { CRYPTO 2014, Part I*, volume 8616 of *Lecture Notes in Computer Science*, pages 480{499, Santa Barbara, CA, USA, August 17{21, 2014. Springer, Heidelberg, Germany. 77

[Can17] Francesco Paolo Cantelli. Sulla probabilita come limite della frequenza. *Atti Accad. Naz. Lincei*, 26(1):39{45, 1917. 11

[Can97] Ran Canetti. Towards realizing random oracles: Hash functions that hide all partial information. In Burton S. Kaliski Jr., editor, *Advances in Cryptology { CRYPTO'97*, volume 1294 of *Lecture Notes in Computer Science*, pages 455{469, Santa Barbara, CA, USA, August 17{21, 1997. Springer, Heidelberg, Germany. 31

[CD08] Ran Canetti and Ronny Ramzi Dakdouk. Obfuscating point functions with multibit output. In Nigel P. Smart, editor, *Advances in Cryptology { EUROCRYPT 2008*, volume 4965 of *Lecture Notes in Computer Science*, pages 489{508, Istanbul, Turkey, April 13{17, 2008. Springer, Heidelberg, Germany. 1, 31

[CGH04] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited. *J. ACM*, 51(4):557{594, July 2004. 4, 31

[CGH+15] Jean-Sebastien Coron, Craig Gentry, Shai Halevi, Tancrede Lepoint, Hemanta K. Maji, Eric Miles, Mariana Raykova, Amit Sahai, and Mehdi Tibouchi. Zeroizing without low-level zeroes: New MMAP attacks and their limitations. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *Advances in Cryptology { CRYPTO 2015, Part I*, volume 9215 of *Lecture Notes*

in Computer Science, pages 247{266, Santa Barbara, CA, USA, August 16{20, 2015. Springer, Heidelberg, Germany. 2

[CGP15] Ran Canetti, Sha Goldwasser, and Oxana Poburinnaya. Adaptively secure two-party computation from indistinguishability obfuscation. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015: 12th Theory of Cryptography Conference, Part II*, volume 9015 of *Lecture Notes in Computer Science*, pages 557{585, Warsaw, Poland, March 23{25, 2015. Springer, Heidelberg, Germany. 2

[CHL+15] Jung Hee Cheon, Kyoohyung Han, Changmin Lee, Hansol Ryu, and Damien Stehle. Cryptanalysis of the multilinear map over the integers. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology { EUROCRYPT 2015, Part I*, volume 9056 of *Lecture Notes in Computer Science*, pages 3{12, So a, Bulgaria, April 26{30, 2015. Springer, Heidelberg, Germany. 2

[CKP15] Ran Canetti, Yael Tauman Kalai, and Omer Paneth. On obfuscation with random oracles. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015: 12th Theory of Cryptography Conference, Part II*, volume 9015 of *Lecture Notes in Computer Science*, pages 456{467, Warsaw, Poland, March 23{25, 2015. Springer, Heidelberg, Germany. 4, 6, 31, 32, 33, 34, 35, 37, 38, 40, 56, 92, 106, 107, 118, 119, 135

[CLLT16] Jean-Sebastien Coron, Moon Sung Lee, Tancrede Lepoint, and Mehdi Tibouchi. Cryptanalysis of GGH15 multilinear maps. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology { CRYPTO 2016, Part II*, volume 9815 of *Lecture Notes in Computer Science*, pages 607{628, Santa Barbara, CA, USA, August 14{18, 2016. Springer, Heidelberg, Germany. 2

[CLMP13] Kai-Min Chung, Huijia Lin, Mohammad Mahmoody, and Rafael Pass. On the power of nonuniformity in proofs of security. In Robert D. Kleinberg, editor, *ITCS 2013: 4th Innovations in Theoretical Computer Science*, pages 389{400, Berkeley, CA, USA, January 9{12, 2013. Association for Computing Machinery. 28

[CLT13] Jean-Sebastien Coron, Tancrede Lepoint, and Mehdi Tibouchi. Practical multilinear maps over the integers. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology { CRYPTO 2013, Part I*, volume 8042 of *Lecture Notes in Computer Science*, pages 476{493, Santa Barbara, CA, USA, August 18{22, 2013. Springer, Heidelberg, Germany. 1

[CLTV15] Ran Canetti, Huijia Lin, Stefano Tessaro, and Vinod Vaikuntanathan. Obfuscation of probabilistic circuits and applications. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015: 12th Theory of Cryptography Conference, Part II*, volume 9015 of *Lecture Notes in Computer Science*, pages

468{497, Warsaw, Poland, March 23{25, 2015. Springer, Heidelberg, Germany. 79

[CM14] Michael Clear and Ciaran McGoldrick. Bootstrappable identity-based fully homomorphic encryption. In Dimitris Gritzalis, Aggelos Kiayias, and Ioannis G. Askoxylakis, editors, *CANS 14: 13th International Conference on Cryptology and Network Security*, volume 8813 of *Lecture Notes in Computer Science*, pages 1{19, Heraklion, Crete, Greece, October 22{24, 2014. Springer, Heidelberg, Germany. 2

[CMR98] Ran Canetti, Daniele Micciancio, and Omer Reingold. Perfectly one-way probabilistic hash functions (preliminary version). In *30th Annual ACM Symposium on Theory of Computing*, pages 131{140, Dallas, TX, USA, May 23{26, 1998. ACM Press. 1, 31

[CRV10] Ran Canetti, Guy N. Rothblum, and Mayank Varia. Obfuscation of hyperplane membership. In Daniele Micciancio, editor, *TCC 2010: 7th Theory of Cryptography Conference*, volume 5978 of *Lecture Notes in Computer Science*, pages 72{89, Zurich, Switzerland, February 9{11, 2010. Springer, Heidelberg, Germany. 1, 31

[CYG+15] Rong Cheng, Jingbo Yan, Chaowen Guan, Fangguo Zhang, and Kui Ren. Verifiable searchable symmetric encryption from indistinguishability obfuscation. In *Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security*, ASIA CCS '15, pages 621{626, New York, NY, USA, 2015. ACM. 2

[DDN00] Danny Dolev, Cynthia Dwork, and Moni Naor. Non-malleable cryptography. In *SIAM Journal on Computing*, pages 542{552, 2000. 2

[DHRW16] Yevgeniy Dodis, Shai Halevi, Ron D. Rothblum, and Daniel Wichs. Spooky encryption and its applications. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology { CRYPTO 2016, Part III*, volume 9816 of *Lecture Notes in Computer Science*, pages 93{122, Santa Barbara, CA, USA, August 14{18, 2016. Springer, Heidelberg, Germany. 20, 21

[DS05] Yevgeniy Dodis and Adam Smith. Correcting errors without leaking partial information. In Harold N. Gabow and Ronald Fagin, editors, *37th Annual ACM Symposium on Theory of Computing*, pages 654{663, Baltimore, MA, USA, May 22{24, 2005. ACM Press. 31

[DTT10] Anindya De, Luca Trevisan, and Madhur Tulsiani. Time space tradeoffs for attacks against one-way functions and PRGs. In Tal Rabin, editor, *Advances in Cryptology { CRYPTO 2010*, volume 6223 of *Lecture Notes in Computer Science*, pages 649{665, Santa Barbara, CA, USA, August 15{19, 2010. Springer, Heidelberg, Germany. 117, 165

[FFS88] Uriel Feige, Amos Fiat, and Adi Shamir. Zero-knowledge proofs of identity. *Journal of Cryptology*, 1(2):77{94, 1988. 80

[FS87] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identi cation and signature problems. In Andrew M. Odlyzko, editor, *Advances in Cryptology { CRYPTO'86*, volume 263 of *Lecture Notes in Computer Science*, pages 186{194, Santa Barbara, CA, USA, August 1987. Springer, Heidelberg, Germany. 80

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In Michael Mitzenmacher, editor, *41st Annual ACM Symposium on Theory of Computing*, pages 169{178, Bethesda, MD, USA, May 31 { June 2, 2009. ACM Press. 6, 77, 78, 87

[GGH13a] Sanjam Garg, Craig Gentry, and Shai Halevi. Candidate multilinear maps from ideal lattices. In Thomas Johansson and Phong Q. Nguyen, editors, *Advances in Cryptology { EUROCRYPT 2013*, volume 7881 of *Lecture Notes in Computer Science*, pages 1{17, Athens, Greece, May 26{30, 2013. Springer, Heidelberg, Germany. 1

[GGH+13b] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *54th Annual Symposium on Foundations of Computer Science*, pages 40{49, Berkeley, CA, USA, October 26{29, 2013. IEEE Computer Society Press. 1, 2, 79

[GGH15] Craig Gentry, Sergey Gorbunov, and Shai Halevi. Graph-induced multilinear maps from lattices. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015: 12th Theory of Cryptography Conference, Part II*, volume 9015 of *Lecture Notes in Computer Science*, pages 498{527, Warsaw, Poland, March 23{25, 2015. Springer, Heidelberg, Germany. 1

[GGHR14] Sanjam Garg, Craig Gentry, Shai Halevi, and Mariana Raykova. Two-round secure MPC from indistinguishability obfuscation. In Yehuda Lindell, editor, *TCC 2014: 11th Theory of Cryptography Conference*, volume 8349 of *Lecture Notes in Computer Science*, pages 74{94, San Diego, CA, USA, February 24{26, 2014. Springer, Heidelberg, Germany. 2

[GGSW13] Sanjam Garg, Craig Gentry, Amit Sahai, and Brent Waters. Witness encryption and its applications. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th Annual ACM Symposium on Theory of Computing*, pages 467{476, Palo Alto, CA, USA, June 1{4, 2013. ACM Press. 1, 6, 16, 95, 96

[GK05] Sha Goldwasser and Yael Tauman Kalai. On the impossibility of obfuscation with auxiliary input. In *46th Annual Symposium on Foundations of Computer*

Science, pages 553{562, Pittsburgh, PA, USA, October 23{25, 2005. IEEE Computer Society Press. 1, 6, 31

[GKLM12] Vipul Goyal, Virendra Kumar, Satyanarayana V. Lokam, and Mohammad Mahmoody. On black-box reductions between predicate encryption schemes. In Ronald Cramer, editor, *TCC 2012: 9th Theory of Cryptography Conference*, volume 7194 of *Lecture Notes in Computer Science*, pages 440{457, Taormina, Sicily, Italy, March 19{21, 2012. Springer, Heidelberg, Germany. 39, 55

[GKP+13] Sha Goldwasser, Yael Tauman Kalai, Raluca A. Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich. Reusable garbled circuits and succinct functional encryption. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th Annual ACM Symposium on Theory of Computing*, pages 555{564, Palo Alto, CA, USA, June 1{4, 2013. ACM Press. 152, 154

[GL89] Oded Goldreich and Leonid A. Levin. A hard-core predicate for all one-way functions. In *21st Annual ACM Symposium on Theory of Computing*, pages 25{32, Seattle, WA, USA, May 15{17, 1989. ACM Press. 103, 131

[GMM+16] Sanjam Garg, Eric Miles, Pratyay Mukherjee, Amit Sahai, Akshayaram Srinivasan, and Mark Zhandry. Secure obfuscation in a weak multilinear map model. Cryptology ePrint Archive, Report 2016/817, 2016. `http://eprint.iacr.org/2016/817`. 2

[GMM17a] Sanjam Garg, Mohammad Mahmoody, and Ameer Mohammed. Lower bounds on obfuscation from all-or-nothing encryption primitives. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology { CRYPTO 2017, Part I*, volume 10401 of *Lecture Notes in Computer Science*, pages 661{695, Santa Barbara, CA, USA, August 20{24, 2017. Springer, Heidelberg, Germany. 5, 6

[GMM17b] Sanjam Garg, Mohammad Mahmoody, and Ameer Mohammed. When does functional encryption imply obfuscation? In Yael Kalai and Leonid Reyzin, editors, *TCC 2017: 15th Theory of Cryptography Conference, Part I*, volume 10677 of *Lecture Notes in Computer Science*, pages 82{115, Baltimore, MD, USA, November 12{15, 2017. Springer, Heidelberg, Germany. 6

[GMR01] Yael Gertner, Tal Malkin, and Omer Reingold. On the impossibility of basing trapdoor functions on trapdoor predicates. In *42nd Annual Symposium on Foundations of Computer Science*, pages 126{135, Las Vegas, NV, USA, October 14{17, 2001. IEEE Computer Society Press. 27

[Gol11] Oded Goldreich. *Basing Non-Interactive Zero-Knowledge on (Enhanced) Trapdoor Permutations: The State of the Art*, pages 406{421. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011. 61

[GP15] Sanjam Garg and Antigoni Polychroniadou. Two-round adaptively secure MPC from indistinguishability obfuscation. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015: 12th Theory of Cryptography Conference, Part II*, volume 9015 of *Lecture Notes in Computer Science*, pages 614{637, Warsaw, Poland, March 23{25, 2015. Springer, Heidelberg, Germany. 2

[GPS16] Sanjam Garg, Omkant Pandey, and Akshayaram Srinivasan. Revisiting the cryptographic hardness of nding a nash equilibrium. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology { CRYPTO 2016, Part II*, volume 9815 of *Lecture Notes in Computer Science*, pages 579{604, Santa Barbara, CA, USA, August 14{18, 2016. Springer, Heidelberg, Germany. 77

[GPSW06] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for ne-grained access control of encrypted data. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *ACM CCS 06: 13th Conference on Computer and Communications Security*, pages 89{ 98, Alexandria, Virginia, USA, October 30 { November 3, 2006. ACM Press. Available as Cryptology ePrint Archive Report 2006/309. 24, 83

[GPSZ17] Sanjam Garg, Omkant Pandey, Akshayaram Srinivasan, and Mark Zhandry. Breaking the sub-exponential barrier in obfustopia. In Jean-Sebastien Coron and Jesper Buus Nielsen, editors, *Advances in Cryptology { EUROCRYPT 2017, Part II*, volume 10211 of *Lecture Notes in Computer Science*, pages 156{181, Paris, France, May 8{12, 2017. Springer, Heidelberg, Germany. 77

[GR07] Sha Goldwasser and Guy N. Rothblum. On best-possible obfuscation. In Salil P. Vadhan, editor, *TCC 2007: 4th Theory of Cryptography Conference*, volume 4392 of *Lecture Notes in Computer Science*, pages 194{213, Amsterdam, The Netherlands, February 21{24, 2007. Springer, Heidelberg, Germany. 1, 6, 97

[GS01] Geo rey Grimmett and David Stirzaker. *Probability and random processes*. Oxford university press, 2001. 11

[GS02] Craig Gentry and Alice Silverberg. Hierarchical ID-based cryptography. In Yuliang Zheng, editor, *Advances in Cryptology { ASIACRYPT 2002*, volume 2501 of *Lecture Notes in Computer Science*, pages 548{566, Queenstown, New Zealand, December 1{5, 2002. Springer, Heidelberg, Germany. 62

[GT00] Rosario Gennaro and Luca Trevisan. Lower bounds on the e ciency of generic cryptographic constructions. In *41st Annual Symposium on Foundations of Computer Science*, pages 305{313, Redondo Beach, CA, USA, November 12{ 14, 2000. IEEE Computer Society Press. 28, 67

[GW11] Craig Gentry and Daniel Wichs. Separating succinct non-interactive arguments from all falsi able assumptions. In Lance Fortnow and Salil P. Vadhan, editors, *43rd Annual ACM Symposium on Theory of Computing*, pages 99{108, San Jose, CA, USA, June 6{8, 2011. ACM Press. 8, 63

[Had00] Satoshi Hada. Zero-knowledge and code obfuscation. In Tatsuaki Okamoto, editor, *Advances in Cryptology { ASIACRYPT 2000*, volume 1976 of *Lecture Notes in Computer Science*, pages 443{457, Kyoto, Japan, December 3{7, 2000. Springer, Heidelberg, Germany. 1

[HJ16] Yupu Hu and Huiwen Jia. Cryptanalysis of GGH map. In Marc Fischlin and Jean-Sebastien Coron, editors, *Advances in Cryptology { EUROCRYPT 2016, Part I*, volume 9665 of *Lecture Notes in Computer Science*, pages 537{565, Vienna, Austria, May 8{12, 2016. Springer, Heidelberg, Germany. 2

[HL02] Jeremy Horwitz and Ben Lynn. Toward hierarchical identity-based encryption. In Lars R. Knudsen, editor, *Advances in Cryptology { EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 466{481, Amsterdam, The Netherlands, April 28 { May 2, 2002. Springer, Heidelberg, Germany. 33, 62

[Hol06] Thomas Holenstein. Strengthening key agreement using hard-core sets - PhD thesis, 2006. 62, 69, 70

[Hol15] Thomas Holenstein. Complexity theory, 2015. `http://www.complexity.ethz.ch/education/Lectures/ComplexityFS15/skript_printable.pdf`. 12

[HR04] Chun-Yuan Hsiao and Leonid Reyzin. Finding collisions on a public road, or do secure hash functions need secret coins? In Matthew Franklin, editor, *Advances in Cryptology { CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 92{105, Santa Barbara, CA, USA, August 15{19, 2004. Springer, Heidelberg, Germany. 27

[HRsV07] Susan Hohenberger, Guy N. Rothblum, abhi shelat, and Vinod Vaikuntanathan. Securely obfuscating re-encryption. In Salil P. Vadhan, editor, *TCC 2007: 4th Theory of Cryptography Conference*, volume 4392 of *Lecture Notes in Computer Science*, pages 233{252, Amsterdam, The Netherlands, February 21{24, 2007. Springer, Heidelberg, Germany. 1, 31

[IR89] Russell Impagliazzo and Steven Rudich. Limits on the provable consequences of one-way permutations. In *21st Annual ACM Symposium on Theory of Computing*, pages 44{61, Seattle, WA, USA, May 15{17, 1989. ACM Press. 2, 3, 5, 9, 26, 27, 28, 32, 62, 67, 69, 78, 80, 153, 155

[Jou00] Antoine Joux. A one round protocol for tripartite di e{hellman. In Wieb Bosma, editor, *Algorithmic Number Theory*, pages 385{393, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg. 62

[Lin16] Huijia Lin. Indistinguishability obfuscation from constant-degree graded encoding schemes. In Marc Fischlin and Jean-Sebastien Coron, editors, *Advances in Cryptology { EUROCRYPT 2016, Part I*, volume 9665 of *Lecture Notes in Computer Science*, pages 28{57, Vienna, Austria, May 8{12, 2016. Springer, Heidelberg, Germany. 2, 79, 153

[Lin17] Huijia Lin. Indistinguishability obfuscation from SXDH on 5-linear maps and locality-5 PRGs. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology { CRYPTO 2017, Part I*, volume 10401 of *Lecture Notes in Computer Science*, pages 599{629, Santa Barbara, CA, USA, August 20{24, 2017. Springer, Heidelberg, Germany. 2

[LPS04] Ben Lynn, Manoj Prabhakaran, and Amit Sahai. Positive results and techniques for obfuscation. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology { EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 20{39, Interlaken, Switzerland, May 2{6, 2004. Springer, Heidelberg, Germany. 1, 31, 32

[LTV12] Adriana Lopez-Alt, Eran Tromer, and Vinod Vaikuntanathan. On-the- y multiparty computation on the cloud via multikey fully homomorphic encryption. In Howard J. Karlo and Toniann Pitassi, editors, *44th Annual ACM Symposium on Theory of Computing*, pages 1219{1234, New York, NY, USA, May 19{22, 2012. ACM Press. 20, 21

[LV16] Huijia Lin and Vinod Vaikuntanathan. Indistinguishability obfuscation from DDH-like assumptions on constant-degree graded encodings. In Irit Dinur, editor, *57th Annual Symposium on Foundations of Computer Science*, pages 11{20, New Brunswick, NJ, USA, October 9{11, 2016. IEEE Computer Society Press. 2

[McC90] Kevin S. McCurley. The discrete logarithm problem. In *Proc. of the AMS Symposia in Applied Mathematics: Computational Number Theory and Cryptography*, pages 49{74. American Mathematical Society, 1990. 36, 37, 40

[MMN16a] Mohammad Mahmoody, Ameer Mohammed, and Soheil Nematihaji. On the impossibility of virtual black-box obfuscation in idealized models. In Eyal Kushilevitz and Tal Malkin, editors, *TCC 2016-A: 13th Theory of Cryptography Conference, Part I*, volume 9562 of *Lecture Notes in Computer Science*, pages 18{48, Tel Aviv, Israel, January 10{13, 2016. Springer, Heidelberg, Germany. 4

[MMN+16b] Mohammad Mahmoody, Ameer Mohammed, Soheil Nematihaji, Rafael Pass, and Abhi Shelat. Lower bounds on assumptions behind indistinguishability obfuscation. In Eyal Kushilevitz and Tal Malkin, editors, *TCC 2016-A: 13th Theory of Cryptography Conference, Part I*, volume 9562 of *Lecture Notes in Computer Science*, pages 49{66, Tel Aviv, Israel, January 10{13, 2016. Springer, Heidelberg, Germany. 6

[MP12] Mohammad Mahmoody and Rafael Pass. The curious case of non-interactive commitments - on the power of black-box vs. non-black-box use of primitives. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology { CRYPTO 2012*, volume 7417 of *Lecture Notes in Computer Science*, pages 701{718, Santa Barbara, CA, USA, August 19{23, 2012. Springer, Heidelberg, Germany. 61, 67, 68

[MSW14] Eric Miles, Amit Sahai, and Mor Weiss. Protecting obfuscation against arithmetic attacks. Cryptology ePrint Archive, Report 2014/878, 2014. `http://eprint.iacr.org/2014/878`. 2

[MSZ16] Eric Miles, Amit Sahai, and Mark Zhandry. Annihilation attacks for multilinear maps: Cryptanalysis of indistinguishability obfuscation over GGH13. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology { CRYPTO 2016, Part II*, volume 9815 of *Lecture Notes in Computer Science*, pages 629{658, Santa Barbara, CA, USA, August 14{18, 2016. Springer, Heidelberg, Germany. 2

[MW16] Pratyay Mukherjee and Daniel Wichs. Two round multiparty computation via multi-key FHE. In Marc Fischlin and Jean-Sebastien Coron, editors, *Advances in Cryptology { EUROCRYPT 2016, Part II*, volume 9666 of *Lecture Notes in Computer Science*, pages 735{763, Vienna, Austria, May 8{12, 2016. Springer, Heidelberg, Germany. 20, 21

[MX10] M. Mahmoody and D. Xiao. On the power of randomized reductions and the checkability of sat. In *2010 IEEE 25th Annual Conference on Computational Complexity*, pages 64{75, June 2010. 62

[Nao03] Moni Naor. On cryptographic assumptions and challenges (invited talk). In Dan Boneh, editor, *Advances in Cryptology { CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 96{109, Santa Barbara, CA, USA, August 17{21, 2003. Springer, Heidelberg, Germany. 63

[Pas11] Rafael Pass. Limits of provable security from standard assumptions. In Lance Fortnow and Salil P. Vadhan, editors, *43rd Annual ACM Symposium on Theory of Computing*, pages 109{118, San Jose, CA, USA, June 6{8, 2011. ACM Press. 63

[PS16] Rafael Pass and Abhi Shelat. Impossibility of VBB obfuscation with ideal constant-degree graded encodings. In Eyal Kushilevitz and Tal Malkin, editors, *TCC 2016-A: 13th Theory of Cryptography Conference, Part I*, volume 9562 of *Lecture Notes in Computer Science*, pages 3{17, Tel Aviv, Israel, January 10{ 13, 2016. Springer, Heidelberg, Germany. 4, 6, 32, 33, 34, 35, 36, 37, 43, 49, 50

[PST14] Rafael Pass, Karn Seth, and Sidharth Telang. Indistinguishability obfuscation from semantically-secure multilinear encodings. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology { CRYPTO 2014, Part I*, volume 8616 of *Lecture Notes in Computer Science*, pages 500{517, Santa Barbara, CA, USA, August 17{21, 2014. Springer, Heidelberg, Germany. 2, 63

[PTV11] Rafael Pass, Wei-Lung Dustin Tseng, and Muthuramakrishnan Venkitasubramaniam. Towards non-black-box lower bounds in cryptography. In Yuval Ishai, editor, *TCC 2011: 8th Theory of Cryptography Conference*, volume 6597 of *Lecture Notes in Computer Science*, pages 579{596, Providence, RI, USA, March 28{30, 2011. Springer, Heidelberg, Germany. 8

[RAD78] Ronald L. Rivest, Len Adleman, and Michael L. Dertouzos. On data banks and privacy homomorphisms. In Richard A. DeMillo, David P. Dobkin, Anita K. Jones, and Richard J. Lipton, editors, *Foundations of Secure Computation*, pages 165{179. Academic Press, 1978. 6

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *37th Annual ACM Symposium on Theory of Computing*, pages 84{93, Baltimore, MA, USA, May 22{24, 2005. ACM Press. 90

[RTV04] Omer Reingold, Luca Trevisan, and Salil P. Vadhan. Notions of reducibility between cryptographic primitives. In Moni Naor, editor, *TCC 2004: 1st Theory of Cryptography Conference*, volume 2951 of *Lecture Notes in Computer Science*, pages 1{20, Cambridge, MA, USA, February 19{21, 2004. Springer, Heidelberg, Germany. 3, 5, 9, 15, 26, 27, 32, 61, 62, 78, 80, 83, 86, 153

[Sah99] Amit Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *40th Annual Symposium on Foundations of Computer Science*, pages 543{553, New York, NY, USA, October 17{19, 1999. IEEE Computer Society Press. 2

[Sho97] Victor Shoup. Lower bounds for discrete logarithms and related problems. In Walter Fumy, editor, *Advances in Cryptology { EUROCRYPT'97*, volume 1233 of *Lecture Notes in Computer Science*, pages 256{266, Konstanz, Germany, May 11{15, 1997. Springer, Heidelberg, Germany. 12, 32, 42, 61

[SW05] Amit Sahai and Brent R. Waters. Fuzzy identity-based encryption. In Ronald Cramer, editor, *Advances in Cryptology { EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 457{473, Aarhus, Denmark, May 22{ 26, 2005. Springer, Heidelberg, Germany. 83

[SW14] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In David B. Shmoys, editor, *46th Annual ACM Symposium on Theory of Computing*, pages 475{484, New York, NY, USA, May 31 { June 3, 2014. ACM Press. 2, 62, 69, 70, 71, 77

[Wat15] Brent Waters. A punctured programming approach to adaptively secure functional encryption. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *Advances in Cryptology { CRYPTO 2015, Part II*, volume 9216 of *Lecture Notes in Computer Science*, pages 678{697, Santa Barbara, CA, USA, August 16{20, 2015. Springer, Heidelberg, Germany. 1

[Wee05] Hoeteck Wee. On obfuscating point functions. In Harold N. Gabow and Ronald Fagin, editors, *37th Annual ACM Symposium on Theory of Computing*, pages 523{532, Baltimore, MA, USA, May 22{24, 2005. ACM Press. 1, 31

[Zim15] Joe Zimmerman. How to obfuscate programs directly. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology { EUROCRYPT 2015, Part II*, volume 9057 of *Lecture Notes in Computer Science*, pages 439{467, So a, Bulgaria, April 26{30, 2015. Springer, Heidelberg, Germany. 2