Lower Bounds on Obfuscation from All-or-Nothing Encryption Primitives

Sanjam Garg[1,⋆], Mohammad Mahmoody[2,⋆⋆], and Ameer Mohammed[2,⋆⋆⋆]

[1] UC Berkeley, sanjamg@berkeley.edu
[2] University of Virginia, {mohammad,ameer}@virginia.edu

Abstract. Indistinguishability obfuscation (IO) enables many heretofore out-of-reach applications in cryptography. However, currently all known constructions of IO are based on multilinear maps which are poorly understood. Hence, tremendous research effort has been put towards basing obfuscation on better-understood computational assumptions. Recently, another path to IO has emerged through functional encryption [Anath and Jain, CRYPTO 2015; Bitansky and Vaikuntanathan, FOCS 2015] but such FE schemes currently are still based on multi-linear maps. In this work, we study whether IO could be based on other powerful encryption primitives.

Separations for IO. We show that (assuming that the polynomial hierarchy does not collapse and one-way functions exist) IO cannot be constructed in a black-box manner from powerful all-or-nothing encryption primitives, such as witness encryption (WE), predicate encryption, and fully homomorphic encryption. What unifies these primitives is that they are of the "all-or-nothing" form, meaning either someone has the "right key" in which case they can decrypt the message fully, or they are not supposed to learn anything.

Stronger Model for Separations. One might argue that fully black-box uses of the considered encryption primitives limit their power too much because these primitives can easily lead to non-black-box constructions if the primitive is used in a *self-feeding* fashion — namely, code of the subroutines of the considered primitive could easily be fed as input to the subroutines of the primitive itself. In fact, several important results (e.g., the construction of IO from functional encryption) follow this very recipe. In light of this, we prove our impossibility results with respect to a *stronger* model than the fully black-box framework of Impagliazzo and Rudich (STOC'89) and Reingold, Trevisan, and Vadhan (TCC'04) where the non-black-box technique of self-feeding is actually allowed.

1 Introduction

Program obfuscation provides an extremely powerful tool to make computer programs "unintelligible" while preserving their functionality. Barak, Goldreich, Impagliazzo,

⋆ Research supported in part from DARPA/ARL SAFEWARE Award W911NF15C0210, AFOSR Award FA9550-15-1-0274, NSF CRII Award 1464397, AFOSR YIP Award and research grants by the Okawa Foundation and Visa Inc. The views expressed are those of the author and do not reflect the official policy or position of the funding agencies.

⋆⋆ Supported by NSF CAREER award CCF-1350939.

⋆⋆⋆ Supported by University of Kuwait.

Rudich, Sahai, Vadhan and Yang [11] formulated this notion in various forms and proved that their strongest formulation, called *virtual black-box* (VBB) obfuscation, is impossible for general polynomial size circuits. However, a recent result of Garg, Gentry, Halevi, Raykova, Sahai and Waters [31] presented a candidate construction for a weaker notion of obfuscation, called *indistinguishability obfuscation* (IO). Subsequent work showed that IO, together with one-way functions, enables numerous cryptographic applications making IO a "cryptographic hub" [63].

Since the original work of [31] many constructions of IO were proposed [31,10,18] [3,53,5,65,8,32]. However, all these constructions are based on computational hardness assumptions on multilinear maps [30,27,37]. Going a step further, recent works of Lin [48] and Lin and Vaikunthanatan [49] showed how to weaken the required degree of the employed multilinear maps schemes to be a *constant*. Another line of work showed how to base IO on compact functional encryption [1,13]. However, the current constructions of compact functional encryption are in turn based on IO (or, multilinear maps). In summary, all currently known paths to obfuscation start from multilinear maps, which are poorly understood. In particular, many attacks on the known candidate multilinear map constructions have been shown [30,23,46,25,26,54].

In light of this, it is paramount that we base IO on well-studied computational assumptions. One of the assumptions that has been used in a successful way for realizing sophisticated cryptographic primitives is the Learning with Errors (LWE) assumption [61]. LWE is already known to imply attribute-based encryption [42] (or even predicate encryption [43]), fully homomorphic encryption [36,20,19,38][3], multi-key [24,55,17,60] and spooky homomorphic encryption [29]. One thing that all these primitives share is that they are of an "all-or-nothing" nature. Namely, either someone has the "right" key, in which case they can decrypt the message fully, or if they do not posses a right key, then they are not supposed to learn anything about the plaintext.[4] In this work, our main question is:

> **Main Question:** *Can IO be based on any powerful 'all-or-nothing' encryption primitive such as predicate encryption or fully homomorphic encryption?*

We show that the answer to the above question is essentially "no." However, before stating our results in detail, we stress that we need to be very careful in evaluating impossibility results that relate to such powerful encryption primitives and the framework they are proved in. For example, such a result if proved in the fully black-box framework of [47,62] has limited value as we argue below.[5] Note that the black-box framework restricts to constructions that use the primitive and the adversary (in the security reduction) as a black-box. The reason for being cautious about this framework is that the constructions of powerful encryption primitive offer for a very natural *non*-black-box use. In fact, the construction of IO from compact functional encryption [1,13,2] is

[3] Realizing full-fledged fully-homomorphic encryption needs additional circular security assumptions.

[4] This is in contrast with *functional* encryption where different keys might leak different information about the plaintext.

[5] Such results could still have some value for demonstrating *efficiency* limitations but not for showing infeasibility, as is the goal of this work.

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 [36,63,14,33,34]. Note that the difference 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 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 study in this work. For example, one of the subroutines of predicate encryption 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 an FHE scheme 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 effort to prove a meaningful impossibility result, should aim for proving the result with respect to a more general framework than that of [47,62]. 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 [16] and the more recent works of Asharov and Segev [6,7] 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 [16] 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 [6,7] 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).

In this work, we revisit the models used in [16,6,7] 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 benefit from this "self-feeding" non-black-box technique. For this goal, here we initiate a formal study of a model that we call *extended* black-box, 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 $\mathcal{P}$ 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 $\mathcal{P}$ are predicate encryption, fully homomorphic encryption, etc. An *extended* black-box construction of another primitive $\mathcal{Q}$ (e.g., IO) from $\mathcal{P}$ will be allowed to plant the subroutines of $\mathcal{P}$ inside the circuit C as gates with no further limitations. To be precise, C will be allowed to have oracle gates that call $\mathcal{P}$ itself. Some of major examples of *non-black-box* constructions that fall into this extended model are as follows.

- Gentry's bootstrapping construction [35] plants FHE's own decryption gates inside a circuit that is given as input to the evaluation subroutine. This trick falls into the extended black-box framework since planting gates inside evaluation circuits is allowed.
- The bootstrapping of IO for $\mathbf{NC}_1$ (along with FHE) to obtain IO for $\mathbf{P}/\mathbf{poly}$ [31]. This construction uses $\mathcal{P}$ 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 [4,22] also fall in our framework.
- The construction of IO from functional encryption [1,13,2] 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 extended black-box framework. We note that the constructions of obfuscation based on constant degree graded encodings [48] also fit in our framework.

The above examples show the power of the "fully" extended black-box 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 extended black-box 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 plating 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.

Our Results. The main result of this paper is that several powerful encryption primitives such as predicate encryption and fully-homomorphic encryption are incapable of producing IO via an *extended* black-box construction as described above. A summery of our results is presented in Figure 1. More specifically, we prove the following theorem.

Theorem 1 (Main Result). *Let $\mathcal{P}$ 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 $\mathcal{P}$ in the extended black-box model where one is* allowed *to plant $\mathcal{P}$ gates arbitrarily inside the circuits that are given to $\mathcal{P}$ as input.*

All-or-nothing aspect. One common aspect of *all* of the primitives listed in Theorem 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 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 flavor will also not

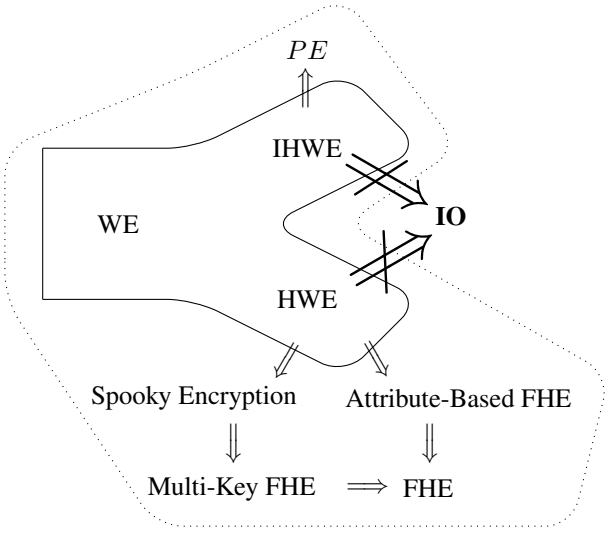

Fig. 1: Summary of our separation results. IHWE denotes Instance Hiding WE and HWE denotes Homomorphic Witness Encryption.

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 LWE? Even though our separations of Theorem 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. In fact, our result only rules out specific paths from LWE toward IO that would go through either of the primitives listed in Theorem 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. Specifically, we consider two (incompatible) variants of WE — namely, instance hiding witness encryption and homomorphic witness encryption. The first 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 extended black-box imply IO. Next, we show that these two primitives extended black-box imply extended versions of all the all-or-nothing primitives listed above. The final separations follow from a specific transitivity lemma that holds in the extended black-box model.

Further Related Work. Now we describe previous work on the complexity of assumptions behind IO and previous works on generalizing the black-box framework of [47,62].

Previous lower bounds on complexity of IO. The work of Mahmoody et. al [52] proved lower bounds on the assumptions that are needed for building IO in a fully black-box way.[6] They showed that, assuming $\mathbf{NP} \neq \mathbf{co\text{-}NP}$, one-way functions or even collision resistant hash functions do not imply IO in a fully black-box way.[7] Relying on the works of [21,58,50] (in the context of VBB obfuscation in idealized models) Mahmoody et. al [52] also showed that building IO from trapdoor permutations or even constant degree graded encoding oracles (constructively) implies that public-key encryption could be based on one-way functions (in a non-black-box way). Therefore, building IO from those primitives would be as hard as basing PKE on OWFs, which is a long standing open question of its own. Relying on the recent beautiful work of Brakerski, Brzuska, and Fleischhacker [15] that ruled out the existence of statistically secure *approximately correct* IO and a variant of Borel-Cantelli lemma, Mahmoody et. al [51] showed how to extend the 'hardness of constructing IO' result of [52] into conditional black-box separations.

Other Non-Black-Box Separations. Proving separations for non-black-box constructions are usually extremely hard. However, there are a few works in this area that we shall discuss here. The work of Baecher, Brzuska, and Fischlin [9] studied various gen-

[6] A previous result of Asharov and Segev [6] proved lower bounds on the complexity of IO *with* oracle gates, which is a *stronger* primitive. (In fact, how this primitive is stronger is tightly related to how we define extensions of primitives. See Section 3 where we formalize the notion of such stronger primitives in a general way.).

[7] Note that since statistically secure IO exists if $\mathbf{P} = \mathbf{NP}$, therefore we need computational assumptions for proving lower bounds for assumptions implying IO.

eralizations of the black-box framework of [62] that also allow some forms of non-black-box use of primitives. The work of Pass, Venkitasubramaniam and Tseng [59] showed that under (new) believable assumptions one can rule out non-black-box constructions of certain cryptographic primitives (e.g., one-way permutations, collision-resistant hash-functions, constant-round statistically hiding commitments) from one-way functions, as long as the security reductions are black-box. Pass [57] showed that the security of some well-known cryptographic protocols and assumptions (e.g., the Schnorr identification scheme) cannot be based on any falsifiable assumptions [56] as long at the security proof is black-box (even if the construction is non-black-box). The work of Genry and Wichs [39] showed that black-box security reductions (together with arbitrary non-black-box constructions) cannot be used to prove the security of any SNARG construction based on any falsifiable cryptographic assumption. Finally, the recent work of Dachman-Soled [28] showed that certain classes of constructions with some limitations, but with specific non-black-box power given to them are not capable of reducing public-key encryption to one way functions.

Organization. Due to limited space, in this draft we only prove the separation of IO from witness encryption (in the extended black-box setting) and refer the reader to the full version of the paper for other separations. In Section 2 we review the needed preliminaries and also review some of the tools that are developed in previous work for proving lower bounds on IO. In Section 3 we discuss the extended black-box model and its relation to extended primitives in detail and give a formal definition of extended black-box constructions from witness encryption. In Section 4 we give a full proof of the extended black-box separation of IO from (even instance-revealing) witness encryption.

2 Preliminaries

Notation. We use "|" 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 $(00|11) = (0011)$, but $(0, 011) \neq (001, 1)$. When writing the probabilities, by putting an algorithm A in the subscript of the probability (e.g., $\Pr_A[\cdot]$) we mean the probability is over A's randomness. We will use n or κ to denote the security parameter. We call an efficient algorithm V a verifier for an $\mathbf{NP}$ relation R if $\mathsf{V}(w, a) = 1$ iff $(w, a) \in R$. We call $L_R = L_{\mathsf{V}} = \{a \mid \exists w, (a, w) \in R\}$ the corresponding $\mathbf{NP}$ language. By PPT we mean a probabilistic polynomial time algorithm. By an *oracle* PPT/algorithm we mean a PPT that might make oracle calls.

2.1 Primitives

In this subsection we define the primitives that we deal with in this work and are defined prior to our work. In the subsequent sections we will define variants of these primitives.

The definition of IO below has 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 [62] where each primitive $\mathcal{Q}$ is simply a class of acceptable functions that we (hope to) efficiently implement given oracle access to functions that implement another primitive $\mathcal{P}$ (see Definition 7).

Definition 2 (Indistinguishability Obfuscation (IO)). *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^κ and outputs a "circuit" B.*
- *Evaluator Ev takes as input (B, x) and outputs y.*

The completeness and soundness conditions assert that:

- *Completeness: For every C, with probability 1 over the randomness of iO, we get $B \leftarrow iO(C, 1^\kappa)$ 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 $\mu(\cdot)$ such that for every two circuits C_0, C_1 that are of the same size and compute the same function, we have:*

$$|\Pr_{iO}[D(iO(C_0, 1^\kappa) = 1] - \Pr_{iO}[D(iO(C_1, 1^\kappa) = 1]| \leq \mu(\kappa)$$

Definition 3 (Approximate IO). *For function $0 < \epsilon(n) \leq 1$, an ϵ-approximate IO scheme is defined similarly to an IO scheme with a relaxed completeness condition:*

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

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

Definition 4 (Witness Encryption (WE) indexed by verifier V). *Let L be an* **NP** *language with a corresponding efficient relation verifier V (that takes instance x and witness w and either accepts or rejects). A* witness encryption *scheme for relation defined by V consists of two PPT algorithms $(\mathrm{Enc}, \mathrm{Dec}_\mathsf{V})$ defined as follows:*

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

We also need the following completeness and security properties:

- ***Completeness:** For any security parameter κ, any (a, w) such that $\mathsf{V}(a, w) = 1$, and any m it holds that*

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

- ***Security:** For any PPT adversary A, there exists a negligible function $\mu(.)$ such that for all $a \notin L_\mathsf{V}$ (i.e., that there is no w for which $\mathsf{V}(a, w) = 1$) and any $m_0 \neq m_1$ of the same length $|m_0| = |m_1|$ the following holds:*

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

When we talk about the witness encryption as a primitive (not an indexed family) we refer to the special case of the 'complete' verifier V *which is a circuit evaluation algorithm and* $\mathsf{V}(w, a) = 1$ *if* $a(w) = 1$ *where* a *is a circuit evaluated on witness* w*.*

The family version of WE in Definition 4 allows the verifier V to be part of the definition of the primitive. However, the standard notion of WE uses the "universal" V which allows us to obtain WE for any other efficient relation verifier V.

The following variant of witness encryption strengthens the functionality.

Definition 5 (Instance-revealing Witness Encryption (IRWE)). *A witness encryption scheme is said to be* instance-revealing *if it satisfies the properties of Definition 4 and, in addition, includes the following subroutine.*

- ***Instance-Revealing Functionality****:* $\mathrm{Rev}(c)$ *given ciphertext* c *outputs* $a \in \{0,1\}^s \cup \{\bot\}$*, and for every* a, m, κ*:*

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

2.2 Black-Box Constructions and Separations

Impagliazzo and Rudich [47] 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 [62]. The work of Baecher, Brzuska, and Fischlin [9] further studied the black-box framework and studied variants of the definition of black-box constructions. We first start by recalling the definition of cryptographic primitives, and then will go over the notion of (fully) black-box constructions.

Definition 6 (Cryptographic Primitives [62]). *A* primitive $\mathcal{P} = (\mathcal{F}, \mathcal{R})$ *is defined as set of functions* $\mathcal{F}$ *and a relation* $\mathcal{R}$ *between functions. A (possibly inefficient) function* $F \in \{0,1\}^* \to \{0,1\}^*$ *is a correct implementation of* $\mathcal{P}$ *if* $F \in \mathcal{F}$*, and a (possibly inefficient) adversary* A *breaks an implementation* $F \in \mathcal{F}$ *if* $(A, F) \in \mathcal{R}$*.*

Definition 7 (black-box constructions [62]). *A* black-box *construction of a primitive* $\mathcal{Q}$ *from a primitive* $\mathcal{P}$ *consists of two PPT algorithms* (Q, S)*:*

1. *Implementation: For any oracle* P *that implements* $\mathcal{P}$*,* Q^P *implements* $\mathcal{Q}$*.*
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*.*

Definition 8 (Black-box constructions of IO). *A fully-black-box construction of IO from any primitive* $\mathcal{P}$ *could be defined by combining Definitions 7 and 2.*

The issue of oracles gates. Note that in any such construction of Definition 8 the input circuits to the obfuscation subroutine do not have any oracle gates in them, while the obfuscation algorithm and the evaluation procedure are allowed to use the oracle implementing $\mathcal{P}$. In Section 3 we will see that one can also define an *extended* variant of the IO primitive (as it was done in [6,7]) in which the input circuits have oracle gates.

2.3 Black-Box Separations

In this section we recall lemmas that can be used for proving black-box impossibility results (a.k.a. separations). The arguments described in this section are borrowed from a collection of recent works [21,50,58,52,15,51] where a framework for proving lower bounds for (assumptions behind) IO are laid out. However, the focus in those works was to prove lower bounds for IO in the (standard) black-box model rather than the extended model. We will indeed use those tools/lemmas by relating the extended black-box model to the black-box model.

Idealized models/oracles and probability measures over them. An idealized model $\mathcal{I}$ is a randomized oracle that supposedly implements a primitive (with high probability over the choice of oracle); examples include the random oracle, random trapdoor permutation oracle, generic group model, graded encoding model, etc. An $I \leftarrow \mathcal{I}$ can (usually) be represented as a sequence $(I_1, I_2, \dots)$ of *finite* random variables, where I_n is the description of the prefix of I that is defined for inputs whose length is parameterized by (a function of) n. The measure over the actual infinite sample $I \leftarrow \mathcal{I}$ could be defined through the given finite distributions $\mathcal{D}_i$ over I_i.[8]

Definition 9 (Oracle-fixed constructions in idealized models [52]). *We say a primitive $\mathcal{P}$ has an* oracle-fixed *construction in idealized model $\mathcal{I}$ if there is an oracle-aided algorithm P such that:*

- **Completeness:** *P^I implements $\mathcal{P}$ correctly for every $I \leftarrow \mathcal{I}$.*
- **Black-box security:** *Let A be an oracle-aided adversary $A^{\mathcal{I}}$ where the* query complexity *of A is bounded by the specified complexity of the attacks for primitive $\mathcal{P}$. For example if $\mathcal{P}$ is polynomially secure (resp., quasi-polynomially secure), then A only asks a polynomial (resp., quasi-polynomial) number of queries but is computationally unbounded otherwise. Then, for any such A, with measure one over the choice of $I \xleftarrow{s} \mathcal{I}$, it holds that A does* not *break P^I.*[9]

Definition 10 (Oracle-mixed constructions in idealized models [52]). *An* oracle-mixed *construction of a primitive $\mathcal{P}$ in idealized model $\mathcal{I}$ is defined similarly to the oracle-fixed definition, but with the difference that the correctness and soundness conditions of the construction $P^{\mathcal{I}}$ hold when the probabilities are taken over $I \leftarrow \mathcal{I}$ as well.*

Lemma 11 (Composition lemma [52]). *Suppose Q is a fully-black-box construction of primitive $\mathcal{Q}$ from primitive $\mathcal{P}$, and suppose P is an oracle-fixed construction for primitive $\mathcal{P}$ relative to $\mathcal{I}$ (according to Definition 10). Then Q^P is an oracle-fixed implementation of $\mathcal{Q}$ relative to the same idealized model $\mathcal{I}$.*

Definition 12 (Oracle-mixed constructions in idealized models [51]). *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:*

[8] Caratheodory's extension theorem shows that such finite probability distributions could always be extended consistently to a measure space over the full infinite space of $I \leftarrow \mathcal{I}$. See Theorem 4.6 of [45] for a proof.

[9] For breaking a primitive, the adversary needs to 'win' with 'sufficient advantage' (this depends on what level of security is needed) over an *infinite* sequence of security parameters.

– **Oracle-Mixed Completeness:** *P^I implements $\mathcal{P}$ correctly where the probabilities are also over $I \leftarrow \mathcal{I}$.*[10] *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 $\mathcal{I}$.*

Using a variant of the Borel–Cantelli lemma, [51] proved that oracle-mixed attacks with *constant* advantage leads to breaking oracle-fixed constructions.

Lemma 13 ([51]). *If there is an algorithm A that oracle-mixed breaks a construction $P^{\mathcal{I}}$ of $\mathcal{P}$ in idealized model $\mathcal{I}$ with advantage $\epsilon(n) \geq \Omega(1)$ for an infinite sequence of security parameters, then the same attacker A oracle-fixed breaks the same construction $P^{\mathcal{I}}$ over a (perhaps more sparse but still) infinite sequence of security parameters.*

The following lemmas follows as a direct corollary to Lemmas 11 and 13.

Lemma 14 (Separation Using Idealized Models). *Suppose $\mathcal{I}$ is an idealized model, and the following conditions are satisfied:*

– **Proving oracle-fixed security of $\mathcal{P}$.** *There is an oracle fixed black-box construction of $\mathcal{P}$ relative to $\mathcal{I}$.*
– **Breaking oracle-mixed security of $\mathcal{Q}$ with $\Omega(1)$ advantage.** *For any construction $Q^{\mathcal{P}}$ of $\mathcal{Q}$ relative to $\mathcal{I}$ there is a computationally-unbounded query-efficient attacker A (whose query complexity is bounded by the level of security demanded by $\mathcal{P}$) such that for an infinite sequence of security parameters $n_1 < n_2 < \ldots$ the advantage of A in oracle-mixed breaking $P^{\mathcal{I}}$ is at least $\epsilon(n_i) \geq \Omega(1)$.*

Then there is no fully black-box construction for $\mathcal{Q}$ from $\mathcal{P}$.

2.4 Tools for Getting Black-Box Lower Bounds for IO

The specific techniques for proving separations for IO that is developed in [21,52,15,51] aims at employing Lemma 14 by "compiling" out an idealized oracle $\mathcal{I}$ from an IO construction. Since we know that *statistically* secure IO does not exist in the *plain* model [41] this indicates that perhaps we can compose the two steps and get a query-efficient attacker against IO in the idealized model $\mathcal{I}$. 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 [15] 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.

[10] 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.

Definition 15 (Sub-models). *We call the idealized model/oracle $\mathcal{O}$ a sub-model of the idealized oracle $\mathcal{I}$ with subroutines $(\mathcal{I}_1, \ldots, \mathcal{I}_k)$, denoted by $\mathcal{O} \sqsubseteq \mathcal{I}$, if there is a (possibly empty) $S \subseteq \{1, \ldots, k\}$ such that the idealized oracle $\mathcal{O}$ is sampled as follows:*

- *First sample $I \leftarrow \mathcal{I}$ 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 $S = \varnothing$ then the oracle $\mathcal{O}$ will be empty and we will be back to the plain model.

Definition 16 (Simulatable Compiling Out Procedures for IO). *Suppose $\mathcal{O} \sqsubset \mathcal{I}$. We say that there is a simulatable compiler from IO in idealized model $\mathcal{I}$ into idealized model $\mathcal{O}$ with correctness error ϵ if the following holds. For every implementation $P_{\mathcal{I}} = (iO_{\mathcal{P}}, Ev_{\mathcal{P}})$ of δ-approximate IO in idealized model $\mathcal{I}$ there is a implementation $P_{\mathcal{O}} = (iO_{\mathcal{O}}, Ev_{\mathcal{O}})$ of $(\delta + \epsilon)$-approximate IO in idealized model $\mathcal{O}$ such that the only security requirement for these two implementations is that they are related as follows:*

Simulation: *There is an efficient PPT simulator S and a negligible function $\mu(\cdot)$ such that for any C:*

$$\Delta(S(iO^{\mathcal{I}}(C, 1^\kappa)), iO^{\mathcal{O}}(C, 1^\kappa)) \leq \mu(\kappa)$$

where $\Delta(.,.)$ denotes the statistical distance between random variables.

It is easy to see that the existence of the simulator according to Definition 16 implies that $P_{\mathcal{O}}$ in idealized model $\mathcal{O}$ is "as secure as" $P_{\mathcal{I}}$ in the idealized model $\mathcal{I}$. Namely, any oracle-mixed attacker against the implementation $P_{\mathcal{O}}$ in model $\mathcal{O}$ with advantage δ (over an infinite sequence of security parameters) could be turned in to an attacker against $P_{\mathcal{I}}$ in model $\mathcal{I}$ that breaks against $P_{\mathcal{I}}$ with advantage $\delta - \mathrm{negl}(\kappa)$ over an infinite 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 [15] one can get a general method of breaking IO in idealized models. We first state the result of [15].

Theorem 17 ([15]). *Suppose one-way functions exist, $\mathbf{NP} \not\subseteq \mathbf{coAM}$, and $\delta, \epsilon \colon \mathbb{N} \mapsto [0, 1]$ are such that $2\epsilon(n) + 3\delta(n) < 1 - 1/\mathrm{poly}(n)$, then there is no (ϵ, δ)-approximate statistically-secure IO 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 18 (Attacking IO Using Nested Oracle Compilers). *Suppose $\varnothing = \mathcal{I}_0 \sqsubseteq \mathcal{I}_1 \cdots \sqsubseteq \mathcal{I}_k = \mathcal{I}$ 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 $\mathcal{I}_i$ into model $\mathcal{I}_{i-1}$*

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* $\mathcal{I}$ *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 $P_{\mathcal{I}} = P_{\mathcal{I}_k}$, we iteratively apply the simulatable compiler to get $P_{\mathcal{I}_{i-1}}$ from $P_{\mathcal{I}_i}$ for $i = \{k, ..., 1\}$. Note that the final correctness error that we get is $\epsilon_{\mathcal{I}_0} < k/(100k) < 1/100$, and thus by Theorem 17 there exists a computationally unbounded attacker $A_{\mathcal{I}_0}$ against $P_{\mathcal{I}_0}$ with constant advantage δ. Now, let S_i be the PPT simulator whose existence is guaranteed by Definition 16 for the compiler that transforms $P_{\mathcal{I}_i}$ into $P_{\mathcal{I}_{i-1}}$. We inductively construct an adversary $A_{\mathcal{I}_i}$ against $P_{\mathcal{I}_i}$ from an adversary $A_{\mathcal{I}_{i-1}}$ for $P_{\mathcal{I}_{i-1}}$ starting with $A_{\mathcal{I}_0}$. The construction of $A_{\mathcal{I}_i}$ simply takes its input obfuscation in the $\mathcal{I}_i$ ideal-model $iO^{\mathcal{I}_i}$, runs $S_i(iO^{\mathcal{I}_i})$ and feeds the result to $A_{\mathcal{I}_{i-1}}$ to get its output. Note that, after constant number k, we still get $\delta' < \delta - k\,\mathrm{negl}(\kappa)$ a constant advantage over infinite sequence of security parameters against $P_{\mathcal{I}_k}$.

Finally, by putting Lemma 18 and 14 together we get a lemma for proving black-box lower bounds for IO.

Lemma 19 (Lower Bounds for IO using Oracle Compilers). *Suppose* $\varnothing = \mathcal{I}_0 \sqsubseteq \mathcal{I}_1 \cdots \sqsubseteq \mathcal{I}_k = \mathcal{I}$ *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* $\mathcal{I}_i$ *into model* $\mathcal{I}_{i-1}$ *with correctness error* $\epsilon_i < 1/(100k)$*. If primitive* $\mathcal{P}$ *can be oracle-fixed constructed in the idealized model* $\mathcal{I}$*, then there is no fully black-box construction of IO from* $\mathcal{P}$*.*

We will indeed use Lemma 19 to derive lower bounds for IO even in the *extended* black-box model by relating such constructions to fully black-box constructions.

3 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 [62] and allows certain non-black-box techniques by default. This model uses steps already taken in works of Brakerski, Katz, Segev, and Yerukhimovich [16] and the more recent works of Asharov and Segev [6,7] and takes them to the next level by allowing even non-black-box techniques involving 'self-calls' [1,13,2]. 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 pritmivies inside those circuits. We will define such constructions using the fully black-box framework by first extending these primitives and then allowing the extensions to be used in a black-box way.

We will first give an informal discussion by going over examples of primitives that could be used in an extended black-box way. We then discuss an abstract model that allows formal definitions. We will finally give concrete and formal definitions for the case of witness encryption which is the only primitive that we will formally separate from IO in this draft. For the rest of the separations see the full version of the paper.

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 definition 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 definitions that define 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 verifier, who take as input a circuit C. Whether or not the prover can convince the verifier to accept the interaction depends on the existence of x such that $C(x) = 1$. This definition of the functionality of ZK-Cir-SAT does not depend on the specific 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 $c = \mathrm{Eval}(f, c_1, \ldots, c_k)$ such that decrypting c leads to $f(m_1, \ldots, m_k)$. The correctness definition 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 [64,44]: 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 [12]: 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 verification algorithm P that takes (k, a), interprets k as a circuits 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 $\mathsf{P}(k, \cdot)$ (with k hard coded in it, and a left out as the input) being the "input" circuit KGen subroutine, or alternatively one can think of $\mathsf{P}(\cdot, a)$ (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 definitions 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 $\mathsf{V}(\cdot, a)$ as the circuit given to the Enc algorithm. In this case, the definition of witness encryption (and it security) only depend on the input-output behavior of these 'input circuits' rather their specific 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 different 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 definitions 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 definition of the primitive still extends to such inputs).

3.1 An Abstract Model for Extended Primitives and Constructions

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

Definition 20 (Indexed primitives). *Let $\mathcal{W}$ be a set of (possibly inefficient) functions. An $\mathcal{W}$-*indexed *primitive $\mathcal{P}[\mathcal{W}]$ is indeed a* set *of primitives $\{\mathcal{P}[W]\}_{W\in\mathcal{W}}$ indexed by $W \in \mathcal{W}$ where, for each $W \in \mathcal{W}$, $\mathcal{P}[W] = (\mathcal{F}[W], \mathcal{R}[W])$ is a primitive according to Definition 6.*

For the special case of $\mathcal{W} = \{W\}$ we get back the RTV definition for a primitive.

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 $\mathcal{F}[W]$. We first define restricted forms of indexed primitives then provide various restrictions that will be of interest to us.

Definition 21 (Restrictions of indexed primitives). *For $\mathcal{P}[\mathcal{W}] = \{(\mathcal{F}[W], \mathcal{R}[W])\}_{W\in\mathcal{W}}$ and $\mathcal{P}'[\mathcal{W}'] = \{(\mathcal{F}'[W], \mathcal{R}'[W])\}_{W\in\mathcal{W}'}$, we say $\mathcal{P}'[\mathcal{W}']$ is a* restriction *of $\mathcal{P}[\mathcal{W}]$ if the following two holds. (1) $\mathcal{W}' \subseteq \mathcal{W}$, and (2) for all $W \in \mathcal{W}'$, $\mathcal{F}'[W] \subseteq \mathcal{F}[W]$, and (3) for all $W \in \mathcal{W}'$, $\mathcal{R}'[W] = \mathcal{R}'[W]$.*

Definition 22 (Efficient restrictions). *We call a restriction $\mathcal{P}'[\mathcal{W}']$ of $\mathcal{P}[\mathcal{W}]$ an* efficient *restriction if $\mathcal{W}' = \{w\}$ where w is is a polynomial time algorithm (with no oracle calls). In this case, we call $\mathcal{P}'[w]$ simply a w-restriction of $\mathcal{P}[\mathcal{W}]$.*

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,[11] 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.

[11] 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.

Definition 23 (The universal variant of indexed primitives). *We call $\mathcal{P}'[\{w\}]$ the* universal variant *of $\mathcal{P}[\mathcal{W}]$ if $\mathcal{P}'[\{w\}]$ is an efficient restriction of $\mathcal{P}[\mathcal{W}]$ for the specific algorithm $w(\cdot)$ 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 define *extensions* of universal variants of indexed primitives (i.e., special primitives for short) we need the following definition.

Definition 24 ($w^{(\cdot)}$-restrictions). *For an oracle algorithm $w^{(\cdot)}$ we call $\mathcal{P}'[\mathcal{W}'] = \{(\mathcal{F}'[W], \mathcal{R}[W])\}_{W \in \mathcal{W}'}$ the $w^{(\cdot)}$-restriction of $\mathcal{P}[\mathcal{W}] = \{(\mathcal{F}[W], \mathcal{R}[W])\}_{W \in \mathcal{W}}$, if $\mathcal{P}'[\mathcal{W}']$ is constructed as follows. For all $W \in \mathcal{W}$ and F, we include $W \in \mathcal{W}'$ and $F \in \mathcal{F}'[W]$, if it holds that $W = w^F$ and $F \in \mathcal{F}[W]$.*

Definition 25 (The extended variant of indexed primitives). *We call $\mathcal{P}'[\mathcal{W}']$ the* extended *variant of $\mathcal{P}[\mathcal{W}]$ if $\mathcal{P}'[\mathcal{W}']$ is an $w^{(\cdot)}$-restriction of $\mathcal{P}[\mathcal{W}]$ for the specific $w^{(\cdot)}$ that interprets its input (x, C) as a pair where $C^{(\cdot)}(x)$ is an* oracle-aided *circuit, and then $w(x, C)$ outputs $C^{(\cdot)}(x)$ by forwarding all of C's oracle queries to its own oracle.*

Case of witness encryption. Here we show how to derive the definition 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 first bit and interpret it as a predicate. So WE is indeed indexed by $V \in \mathcal{V}$ which the set of all predicates. Then, the standard definition of witness encryption for circuit satisfiability (which is the most powerful WE among them all) is simply the universal variant of this indexed primitive $\mathrm{WE}[\mathcal{V}]$, and the following will be exactly the definition of the extended universal variant of $\mathrm{WE}[\mathcal{V}]$, which we simply call the extended WE.

In the full version of the paper we give similar definitions for other primitives of predicate encryption, fully homomorphic encrypion, etc.

Definition 26 (Extended Witness Encryption). *Let $\mathsf{V}^{(\mathrm{Enc},\mathrm{Dec})}(w, a)$ be the 'universal circuit-evaluator' Turing machine, which is simply an algorithm with oracle access to $(\mathrm{Enc}, \mathrm{Dec})$ that interprets a as an circuit with possible $(\mathrm{Enc}, \mathrm{Dec})$ gates and runs a on w and forwards any oracle calls made by a to its own orcle and forwards the answer back to the corresponding gate inside a to continue the execusion. An* extended witness encryption *scheme (defined by V) consists of two PPT algorithms $(\mathrm{Enc}, \mathrm{Dec}_{\mathsf{V}})$ defined as follows:*

- $\mathrm{Enc}(a, m, 1^\kappa)$: *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 $c \in \{0,1\}^*$.*
- $\mathrm{Dec}_{\mathsf{V}}(w, c)$: *given ciphertext c and "witness" string w, it either outputs a message $m \in \{0,1\}^*$ or $\perp$.*
- *Correctness and security are defined similarly to Definition 4. But the key point is that here the relation $\mathsf{V}^{(\mathrm{Enc},\mathrm{Dec})}$ is somehow recursively depending on the $(\mathrm{Enc}, \mathrm{Dec} = \mathrm{Dec}_V)$ on smaller input lengths (and so it is well defined).*

3.2 Extended Black-Box Constructions

We are finally ready to define our extended black-box framework. Here we assume that for a primitive $\mathcal{P}$ we have already defined what its extension $\widetilde{\mathcal{P}}$ means.

Definition 27 (Extended Black-Box Constructions – General Case). *Suppose $\mathcal{Q}$ is a primitive and $\widetilde{\mathcal{P}}$ is an extended version of the primitive $\mathcal{P}$. Any fully black-box construction for $\mathcal{Q}$ from $\widetilde{\mathcal{P}}$ (i.e. an extended version of $\mathcal{P}$) is called an* extended black-box *construction of $\mathcal{Q}$ from $\mathcal{P}$.*

Examples. Below are some examples of *non-black-box* constructions in cryptography that fall into the extended black-box framework of Definition 27.

- Gentry's bootstrapping construction [35] plants FHE's own decryption in a circuit for the evaluation subroutine. This trick falls into the extended black-box framework since planting gates inside evaluation circuits is allowed.
- The construction of IO from functional encryption by [1,13] 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 extended black-box framework.

Definition 28 (Formal Definition of Extended Black-Box Constructions from Witness Encryption). *Let $\mathcal{P}$ be witness encryption and $\widetilde{\mathcal{P}}$ be extended witness encryption (Definition 26). Then an extended black-box construction using $\mathcal{P}$ is a fully black-box construction using $\widetilde{\mathcal{P}}$.*

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 29 (Composing extended black-box constructions). *Suppose $\mathcal{P}$, $\mathcal{Q}, \mathcal{R}$ are cryptographic primitives and $\mathcal{Q}, \mathcal{P}$ are special primitive and $\widetilde{\mathcal{Q}}$ is the extended version of $\mathcal{Q}$. If there is an extended black-box construction of $\widetilde{\mathcal{Q}}$ from $\mathcal{P}$ and if there is an extended black-box construction of $\mathcal{R}$ from $\mathcal{Q}$, then there is an extended black-box construction of $\mathcal{R}$ from $\mathcal{P}$.*

Proof. Since there is an extended black-box construction of $\mathcal{R}$ from $\mathcal{Q}$, by Definition 27 it means that there is an extension $\widetilde{\mathcal{Q}}$ of $\mathcal{Q}$ such that there is a fully black-box construction of $\mathcal{R}$ from $\widetilde{\mathcal{Q}}$. On the other hand, again by Definition 27, for any extension of $\mathcal{Q}$, and in particular $\widetilde{\mathcal{Q}}$, there is a fully black-box construction of $\widetilde{\mathcal{Q}}$ from some extension $\widetilde{\mathcal{P}}$ of $\mathcal{P}$. Therefore, since fully-black-box constructions are transitive under nested compositions, there is a fully construction of $\mathcal{R}$ from $\widetilde{\mathcal{P}}$ which (by Definition 27) means that we have an extended black-box construction of $\mathcal{R}$ from $\mathcal{P}$.

Getting more separations. A corollary of Lemma 29 is that if one proves: (a) There is no extended black-box construction of $\mathcal{R}$ from $\mathcal{P}$ and (b) there *is* an extended black-box construction of any extended version $\widetilde{\mathcal{R}}$ (of $\mathcal{R}$) from $\mathcal{Q}$, then these two together imply that: there is no extended black-box construction of $\mathcal{Q}$ from $\mathcal{P}$. We will use this trick to derive our impossibility results from a core of two separations regarding variants of witness encryption. For example, in the full version of the paper we will use this lemma to derive separations between attribute based encryption and IO in the extended black-box model.

4 Separating IO from Instance Revealing Witness Encryption

In this section, we formally prove our first main separation theorem which states that there is no black-box constructions 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 30. *Assume the existence of one-way functions and that* $\mathbf{NP} \not\subseteq \mathbf{coAM}$*. Then there exists no extended black-box 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 29, this would also imply a separation of IO from extended ABE.

Definition 31 (Extended Extractable Instance-Revealing Witness Encryption (ex-EIRWE)). *Let* V *be a universal circuit-evaluator Turing machine as defined in Definition 26. For any given security parameter* κ*, an extended* extractable instance-revealing witness encryption *scheme for* V *consists of three PPT algorithms* $P = (\mathrm{Enc}, \mathrm{Rev}, \mathrm{Dec})$ *defined as follows:*

- $\mathrm{Enc}(a, m, 1^\kappa)$: *given an instance* $a \in \{0,1\}^*$ *and a message* $m \in \{0,1\}^*$*, and security parameter* κ *(and randomness as needed) it outputs* $c \in \{0,1\}^*$.
- $\mathrm{Rev}(c)$: *given ciphertext* c *outputs* $a \in \{0,1\}^* \cup \{\perp\}$.
- $\mathrm{Dec}(w, c)$: *given ciphertext* c *and "witness" string* w*, it outputs a message* $m' \in \{0,1\}^*$.

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

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

$$\Pr_{\mathrm{Enc},\mathrm{Dec}}[\mathrm{Dec}(w, \mathrm{Enc}(a, m, 1^\kappa)) = m] = 1$$

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

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

  Furthermore, for any c *for which there is no* a, m, κ *such that* $\mathrm{Enc}(a, m, 1^\kappa) = c$ *it holds that* $\mathrm{Rev}(c) = \perp$.
- ***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 *of the same and any* $m_0 \neq m_1$ *of the same length* $|m_0| = |m_1|$*, if:*

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

Then:

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

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 32. *Assume the existence of one-way functions and that* $\mathbf{NP} \not\subseteq \mathbf{coAM}$. *Then there exists no* extended *black-box construction of indistinguishability obfuscation from extractable instance-revealing witness encryption for any PPT verification algorithm* V.

Since extended EIRWE implies witness encryption as defined in Definition 4, Theorem 30 trivially follows from Theorem 32, and thus for the remainder of this section we will focus on proving Theorem 32.

4.1 Overview of Proof Techniques

To prove Theorem 32, we will apply Lemma 19 for the idealized extended IRWE model Θ (formally defined in Section 4.2) to prove that there is no black-box construction of IO from any primitive $\mathcal{P}$ that can be oracle-fixed constructed (see Definition 10) from Θ. In particular, we will do so for $\mathcal{P}$ that is the extended EIRWE primitive. Our task is thus twofold: **(1)** to prove that $\mathcal{P}$ can be oracle-fixed constructed from Θ and **(2)** to show a simulatable compilation procedure that compiles out Θ from any IO construction. The first task is proven in Section 4.3 and the second task is proven in Section 4.4. By Lemma 19, this would imply the separation result of IO from $\mathcal{P}$ and prove Theorem 32.

Our oracle, which is more formally defined in Section 4.2, 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 16 in this idealized model.

4.2 The Ideal Model

In this section, we define the distribution of our ideal randomized extended oracle.

Definition 33 (Random Instance-revealing Witness Encryption Oracle). *Let* V *be a universal circuit-evaluator Turing machine (as defined in Definition 26) that takes as input* (w, x) *where* $x = (a, m) \in \{0,1\}^n$ *and outputs* $b \in \{0,1\}$. *We define the following* random instance-revealing witness encryption *(rIRWE) oracle* $\Theta = (\mathrm{Enc}, \mathrm{Rev}, \mathrm{Dec}_{\mathsf{V}})$ *as follows. We specify the sub-oracle* Θ_n *whose inputs are parameterized by* n, *and the actual oracle will be* $\Theta = \{\Theta_n\}_{n\in\mathbb{N}}$.

- $\mathrm{Enc}\colon \{0,1\}^n \mapsto \{0,1\}^{2n}$ *is a random injective function.*
- $\mathrm{Rev}\colon \{0,1\}^{2n} \mapsto \{0,1\}^* \cup \bot$ *is a function that, given an input* $c \in \{0,1\}^{2n}$, *would output the corresponding attribute* a *for which* $\mathrm{Enc}(a, m) = c$. *If there is no such attribute then it outputs* $\bot$ *instead.*
- $\mathrm{Dec}_{\mathsf{V}}\colon \{0,1\}^s \mapsto \{0,1\}^n \cup \{\bot\}$*: Given* $(w, c) \in \{0,1\}^s$, $\mathrm{Dec}(w, c)$ *allows us to decrypt the ciphertext* c *and get* $x = (a, m)$ *as long as the predicate test is satisfied on* (w, a). *More formally, do as follow:*

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

We define a query-answer pair resulting from query q *to subroutine* $T \in \{\mathrm{Enc}, \mathrm{Dec}, \mathrm{Rev}\}$ *with some answer* β *as* $(q \mapsto \beta)_T$*. The oracle* Θ *provides the subroutines for all inputs lengths but, for simplicity, and when* n *is clear from the context, we use* $\Theta = (\mathrm{Enc}, \mathrm{Rev}, \mathrm{Dec}_{\mathsf{V}})$ *to refer to* Θ_n *for a fixed* n*.*

Remark 34. 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*.

4.3 Witness Encryption exists relative to Θ

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

Lemma 35. *There exists a correct and subexponentially-secure oracle-fixed implementation (Definition 10) of extended extractable instance-revealing witness encryption in the ideal* Θ *oracle model.*

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

Definition 36 (Extended Extractable One-way Witness Encryption (ex-EOWE)). *Let* V *be a universal circuit-evaluator Turing machine (as defined in Definition 26) 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* $P = (\mathrm{Enc}, \mathrm{Rev}, \mathrm{Dec}_{\mathsf{V}})$ *defined as follows:*

- $\mathrm{Enc}(a, m, 1^{\kappa})$: *given an instance* $a \in \{0, 1\}^*$*, message* $m \in \{0, 1\}^*$*, and security parameter* κ *(and randomness as needed) it outputs* $c \in \{0, 1\}^*$*.*
- $\mathrm{Rev}(c)$: *given ciphertext* c *returns the underlying attribute* $a \in \{0, 1\}^*$*.*
- $\mathrm{Dec}_{\mathsf{V}}(w, c)$: *given ciphertext* c *and "witness" string* w*, it outputs a message* $m' \in \{0, 1\}^*$*.*

An extended extractable one-way witness encryption scheme satisfies the same correctness properties as Definition 31 but the extractability property is replaced with the following:

– ***Extractable Inversion:*** *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, $k = \text{poly}(\kappa)$, and for all a, if:*

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

Then:

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

Construction 37 (Extended Extractable One-way Witness Encryption) *For any security parameter κ and oracle Θ sampled according to Definition 33, we will implement an extended EOWE scheme P for the universal circuit-evaluator V using $\Theta = (\text{Enc}, \text{Dec}_\mathsf{V})$ as follows:*

– $\text{WEnc}(a, m, 1^\kappa)$: *Given security parameter 1^κ, $a \in \{0,1\}^*$, and message $m \in \{0,1\}^{n/2}$ where $n = 2\max(|a|, \kappa)$, output $\text{Enc}(x)$ where $x = (a, m)$.*
– $\text{WDec}(w, c)$: *Given witness w and ciphertext c, let $x' = \text{Dec}_\mathsf{V}(w, c)$. If $x' \neq \perp$, parse as $x' = (a', m')$ and output m'. Otherwise, output $\perp$.*

Remark 38 (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 is indistinguishability-based, through a simple application of the Goldreich-Levin theorem [40]. Namely, to encrypt a one-bit message b under some attribute a, we would output the ciphertext $c = (\text{Enc}(a, r_1), r_2, \langle r_1, r_2\rangle \oplus b)$ 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 $r_1 = \text{Dec}(w, y_1)$, find the hardcore bit $p = \langle r_1, r_2\rangle$ then output $b = p \oplus y_3$. We obtain the desired indistinguishability security since, by the hardcore-bit security of the original scheme, we have $(\text{Enc}(a, r_1), r_2, \langle r_1, r_2\rangle \oplus 0) \approx (\text{Enc}(a, r_1), r_2, \langle r_1, r_2\rangle \oplus 1)$ for any fixed a.

Lemma 39. *Construction 37 is a correct and subexponentially-secure oracle-fixed implementation (Definition 10) of extended extractable one-way witness encryption in the ideal Θ 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 Θ oracle model) that can invert an encryption of a random message with non-negligible advantage then there exists a (fixed) deterministic straight-line (non-rewinding) extractor E with access to $\Theta = (\text{Enc}, \text{Rev}, \text{Dec}_\mathsf{V})$ that can find 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 $m \xleftarrow{\$} \{0,1\}^k$ uniformly at random then runs $A^\Theta(1^\kappa, c^*)$ where $c^* \leftarrow \text{WEnc}(a, m, 1^\kappa)$ is the challenge. Queries issued by A are handled by E as follows:

– To answer any query $\text{Enc}(x)$ asked by A, it forwards the query to the oracle Θ and returns some answer c.

– To answer any query $\mathrm{Rev}(c)$ asked by A, it forwards the query to the oracle Θ and returns some answer a.
– To answer any query $\mathrm{Dec}_{\mathsf{V}}(w, c)$ asked by A, the extractor first issues a query $\mathrm{Rev}(c)$ to get some answer a. If $a \neq \perp$, it would execute $\mathsf{V}^{\Theta}(w, a)$, forwarding queries asked by V to Θ similar to how it does for A. Finally, it forwards the query $\mathrm{Dec}(w, c)$ to Θ to get some answer x. If $a = \perp$, it returns $\perp$ to A otherwise it returns x.

While handling the queries made by A, if a decryption query $\mathrm{Dec}_{\mathsf{V}}(w, c^*)$ for the challenge ciphertext is issued by A, the extractor will pass this query to Θ, 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 40. *For any PPT adversary A, instances a, if there exists a non-negligible function $\epsilon(.)$ such that:*

$$\Pr\left[A^{\Theta}(1^{\kappa}, c) = m \mid m \xleftarrow{\$} \{0,1\}^{k}, c \leftarrow \mathrm{WEnc}(a, m, 1^{\kappa})\right] \geq \epsilon(\kappa) \tag{1}$$

Then there exists a PPT straight-line extractor E such that:

$$\Pr\left[E^{\Theta, A}(a) = w \wedge \ \mathsf{V}^{\Theta}(w, a) = 1\right] \geq \epsilon(\kappa) - \mathrm{negl}(\kappa) \tag{2}$$

Proof. Let A be an adversary satisfying Equation (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 (2) above). Observe that:

$$\begin{aligned}
\Pr_{\Theta, m}[\mathsf{ExtWin}] &\geq \Pr_{\Theta, m}[\mathsf{ExtWin} \wedge \mathsf{AdvWin}] \\
&= 1 - \Pr_{\Theta, m}[\overline{\mathsf{ExtWin}} \vee \overline{\mathsf{AdvWin}}] \\
&= 1 - \Pr_{\Theta, m}[\overline{\mathsf{ExtWin}} \wedge \mathsf{AdvWin}] - \Pr_{\Theta, m}[\overline{\mathsf{AdvWin}}]
\end{aligned}$$

Since $\Pr[\mathsf{AdvWin}] \geq \epsilon$ for some non-negligible function ϵ, it suffices to show that $\Pr[\overline{\mathsf{ExtWin}} \wedge \mathsf{AdvWin}]$ 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 $\mathrm{Dec}_{\mathsf{V}}(w, c^*)$ for which the answer is $x \neq \perp$ and so the extractor fails to recover the witness. For simplicity of notation define $\mathsf{Win} := \overline{\mathsf{ExtWin}} \wedge \mathsf{AdvWin}$.

We will show that, with overwhelming probability over the choice of oracle Θ, the probability of Win happening is negligible. That is, we will prove the following claim:

Claim. For any negligible function δ, $\Pr_{\Theta}\left[\Pr_{m}[\mathsf{Win}] \geq \sqrt{\delta}\right] \leq \mathrm{negl}(\kappa)$

Proof. Define Bad to be the event that A asks (directly or indirectly) a query of the form $\mathrm{Dec}_{\mathsf{V}}(w, c')$ for some $c' \neq c^*$ for which it has not asked $\mathrm{Enc}(x) = c$ previously. We have that:

$$\Pr_{\Theta, m}[\mathsf{Win}] \leq \Pr_{\Theta, m}[\mathsf{Win} \wedge \overline{\mathsf{Bad}}] + \Pr_{\Theta, m}[\mathsf{Bad}]$$

The probability of Bad over the randomness of Θ 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, $\Pr_{\Theta,m}[\mathsf{Bad}] \leq 1/2^n$.

It remains to show that $\Pr_{\Theta,m}[\mathsf{Win} \wedge \overline{\mathsf{Bad}}]$ 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. Specifically, we show that for any such A that satisfies the event $(\mathsf{Win} \wedge \overline{\mathsf{Bad}})$, there exists another adversary $\widehat{A}$ that depends on A and also satisfies 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 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 $\mathrm{Enc}(x)$ then $\widehat{A}$ forwards the query to Θ to get the answer.
- If A asks a query of the form $\mathrm{Rev}(c)$ then since Bad does not happen, it must be the case that $c = \mathrm{Enc}(a, m)$ 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 $\mathrm{Dec}(w, c^*)$ 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 $\perp$ to A after running $\mathsf{V}^{\Theta}(w, a^*)$ and answering its queries appropriately.
- If A asks a query of the form $\mathrm{Dec}(w, c')$ 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 $\mathrm{Enc}(x') = c'$. Therefore, $\widehat{A}$ would have observed this encryption query and can therefore run $\mathsf{V}^{\Theta}(w, a')$ and return the appropriate answer (x or $\perp$) 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 $\mathrm{Enc}(x^*) = c^*$ 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 $\Pr_{\Theta,m}[\mathsf{Win} \wedge \overline{\mathsf{Bad}}] \leq \delta(\kappa)$ for some negligible δ then $\Pr_{\Theta}[\Pr_m[\mathsf{Win} \wedge \overline{\mathsf{Bad}}] \leq \sqrt{\delta}] \geq 1 - \sqrt{\delta}$, which yields the result.

To conclude the proof of Lemma 40, we can see that the probability that the extractor wins is given by $\Pr[\mathsf{ExtWin}] \geq 1 - \Pr[\overline{\mathsf{ExtWin}} \wedge \mathsf{AdvWin}] - \Pr[\overline{\mathsf{AdvWin}}] \geq \epsilon(\kappa) - \mathrm{negl}(\kappa)$ where ϵ is the non-negligible advantage of the adversary A.

It is clear that Construction 37 is a correct implementation. Furthermore, by Lemma 40, it satisfies the extractability property. Thus, this concludes the proof of Lemma 39.

Proof (of Lemma 35). The existence of extractable instance-revealing witness encryption in the Θ oracle model follows from Lemma 39 and Remark 38.

4.4 Compiling out Θ from IO

In this section, we show a simulatable compiler for compiling out Θ. We adapt the approach outlined in Section 4.1 to the extended ideal IRWE oracle $\Theta = (\mathrm{Enc}, \mathrm{Rev}, \mathrm{Dec}_{\mathsf{V}})$

while making use of Lemma 18, which allows us to compile out Θ in two phases: we first compile out part of Θ 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 [21] 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 41. *Let $R \sqsubseteq \Theta$ be a random oracle where "$\sqsubseteq$" denotes a sub-model relationship (see Definition 15). Then the following holds:*

- *For any IO in the Θ 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.*
- *[21] 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 41 follows directly by [21], 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 42 (Canonical executions). *Web define an oracle algorithm A^Θ relative to rIRWE to be in canonical form if before asking any $\mathrm{Dec}_{\mathsf{V}}(w, c)$ query, A would first get $a \leftarrow \mathrm{Rev}(c)$ then run $\mathsf{V}^\Theta(w, a)$ on its own, making sure to answer any queries of V using Θ. Furthermore, after asking a query $\mathrm{Dec}_{\mathsf{V}}(w, c)$ for which the returned answer is some message $m \neq \perp$, it would ask $\mathrm{Enc}(x)$ 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 43 (Query Types). *For any (not necessarily canonical) oracle algorithm A with access to a rIRWE oracle Θ, we call the queries that are asked by A to Θ as* direct *queries and those queries that are asked by V^Θ 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 Θ. 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 $\mathrm{Dec}_{\mathsf{V}}$ query which prompted V^Θ 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 42, A will run V^Θ before asking $\mathrm{Dec}_{\mathsf{V}}$ 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 Θ 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 42.

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 $\mathrm{Enc}(x)$ **then**
  Set $\rho_q = (x \mapsto R(x))_{\mathrm{Enc}}$
end
if q is a query of the form $\mathrm{Rev}(c)$ **then**
  if $\exists\ (x \mapsto c)_{\mathrm{Enc}} \in Q$ where $x = (a, m)$ **then**
    Set $\rho_q = (c \mapsto a)_{\mathrm{Rev}}$
  else
    Set $\rho_q = (c \mapsto \perp)_{\mathrm{Rev}}$
  end
end
if q is a query of the form $\mathrm{Dec}_{\mathsf{V}}(w, c)$ **then**
  if $\exists\ (x \mapsto c)_{\mathrm{Enc}} \in Q$ **then**
    Initialize $Q_{\mathsf{V}} = \varnothing$ and emulate $b \leftarrow \mathsf{V}^{\Theta}(w, x)$
    for each query q_{V} asked by V **do**
      $\rho_{\mathsf{V}} \leftarrow \texttt{EmulateCall}^{R}(Q \cup Q_{\mathsf{V}}, q_{\mathsf{V}})$
      $Q_{\mathsf{V}} = Q_{\mathsf{V}} \cup \rho_{\mathsf{V}}$
    end
    if $b = 1$ **then**
      Set $\rho_q = ((w, c) \mapsto x)_{\mathrm{Dec}}$
    else
      Set $\rho_q = ((w, c) \mapsto \perp)_{\mathrm{Dec}}$
    end
  else
    Set $\rho_q = ((w, c) \mapsto \perp)_{\mathrm{Dec}}$
  end
end
Return ρ_q

The new obfuscator $\widehat{O}^{R}$ 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 $O = (iO, Ev)$ in the rIRWE oracle model, we construct an $(\delta+\epsilon)$-approximate obfuscator $\widehat{O} = (\widehat{iO}, \widehat{Ev})$ in the random oracle model.

Subroutine $\widehat{iO}^{R}(C)$:

1. *Emulation phase*: Emulate $iO^{\Theta}(C)$. Let T_O be the transcript of this phase and initialize $Q_O := Q(T_O) = \varnothing$. For every query q asked by $iO^{\Theta}(C)$, call $\rho_q \leftarrow \texttt{EmulateCall}^{R}(Q_O, q)$ 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^{Θ} before asking/emulating a Dec query.

2. *Learning phase*: Set $Q_B = \varnothing$ to be the set of query-answer pairs learned during this phase. Set $m = 2\ell_O/\epsilon$ where $\ell_O \leq |iO|$ represents the number of queries asked by iO. Choose $t \xleftarrow{\$} [m]$ uniformly at random then for $i = \{1, ..., t\}$:
   - Choose $z_i \xleftarrow{\$} \{0,1\}^{|C|}$ uniformly at random
   - Run $Ev^{\Theta}(B, z_i)$. For every query q asked by $Ev^{\Theta}(B, z_i)$, call and retrieve $\rho_q \leftarrow \texttt{EmulateCall}^R(Q_O \cup Q_B, q)$ 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 $\widehat{iO}^R(C)$ will be $\widehat{B} = (B, Q_B)$.

Subroutine $\widehat{Ev}^R(\widehat{B}, z)$: To evaluate $\widehat{B} = (B, Q_B)$ on a new random input z we simply emulate $Ev^{\Theta}(B, z)$. For every query q asked by $Ev^{\Theta}(B, z)$, run and set $\rho_q = \texttt{EmulateCall}^R(Q_B, q)$ then add ρ_q to Q_B.

The running time of $\widehat{iO}$. We note that the running time of the new obfuscator $\widehat{iO}$ 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 extended oracle where the PPT V can have oracle gates to subroutines of Θ, we emphasize that since V, which we are executing during `EmulateCall`, is a universal circuit evaluator, its effective running time remains to be a strict polynomial in the size of V and so the issue of exponential or infinite 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 differ when evaluating $\widehat{B}$. Specifically, in the real experiment, $\widehat{Ev}^R(\widehat{B}, z)$ emulates $Ev^{\Theta}(B, z)$ on a random input z and answers any queries by running Q_B, whereas in the ideal experiment, we execute $\widehat{Ev}^R(\widehat{B}, z)$ and answer the queries of $Ev^{\Theta}(B, z)$ using the actual oracle Θ instead. In essence, in the real experiment, we can think of the execution as $Ev^{\widehat{\Theta}}(B, z)$ where $\widehat{\Theta}$ 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 differentiate between the executions $Ev^{\Theta}(B, z)$ and $Ev^{\widehat{\Theta}}(B, z)$.

Let q be a new query that is being asked by $Ev^{\widehat{\Theta}}(B, z)$ and handled by calling $\texttt{EmulateCall}^R(Q_B, q)$. The following are the cases that should be handled:

1. If q is a query of type $\mathrm{Enc}(x)$, then the answer to q will be distributed the same in both experiments.
2. If q is a query of type $\mathrm{Dec}(w, c)$ or $\mathrm{Rev}(c)$ whose answer is determined by Q_B in the real experiment then it is also determined by $Q_O \cup Q_B \supseteq Q_B$ in the ideal experiment and the answers are distributed the same.
3. If q is of type $\mathrm{Dec}(w, c)$ or $\mathrm{Rev}(c)$ 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 $\mathrm{Dec}(w, c)$ 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 $(x \mapsto c)_{\mathrm{Enc}}$ is in $Q_O \setminus Q_B$. 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 $\mathrm{Rev}(c)$ 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 $((a, m) \mapsto c)_{\mathrm{Enc}}$ is in $Q_O \setminus Q_B$. 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 $Ev^{\Theta}(B, x)$ and $Ev^{\widehat{\Theta}}(B, x)$ will be statistically close. More formally, the probability of correctness for $\widehat{iO}$ is:

$$\begin{aligned}\Pr_x[Ev^{\widehat{\Theta}}(B,x) \neq C(x)] &= \Pr_x[Ev^{\widehat{\Theta}}(B,x) \neq C(x) \wedge \neg E(x)] \\ &\quad + \Pr_x[Ev^{\widehat{\Theta}}(B,x) \neq C(x) \wedge E(x)] \\ &\leq \Pr_x[Ev^{\widehat{\Theta}}(B,x) \neq C(x) \wedge \neg E(x)] + \Pr_x[E(x)]\end{aligned}$$

By the approximate functionality of iO, we have that:

$$\Pr_x[iO^{\Theta}(C)(x) \neq C(x)] = \Pr_x[Ev^{\Theta}(B,x) \neq C(x)] \leq \delta(n)$$

Therefore,

$$\Pr_x[Ev^{\widehat{\Theta}}(B,x) \neq C(x) \wedge \neg E(x)] = \Pr_x[Ev^{\Theta}(B,x) \neq C(x) \wedge \neg E(x)] \leq \delta$$

We are thus left to show that $\Pr[E(x)] \leq \epsilon$. 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. $\Pr_x[E(x)] \leq \epsilon$.

Proof. For all $i \in [t]$, let $Q'_{B_i} = Q_{B_i} \cap Q_O$ be the set of query-answer pairs generated by the i'th evaluation $Ev^{\Theta}(B, z_i)$ during the learning phase (Step 2) and are also generated during the obfuscation emulation phase (Step 1). In particular, Q'_{B_i} would

contain the query-answer pairs $((a, m) \mapsto c)_{\mathrm{Enc}}$ 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 $Q'_{B_i} \subseteq Q'_{B_{i+1}}$, 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 finding out the underlying attribute a) via Rev queries during the learning phase.

We say $t \xleftarrow{\$} [m]$ is bad if it is the case that $Q'_{B_t} \neq Q'_{B_{t+1}}$ (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 final evaluation $Q'_{\widehat{B}} := Q'_{B_{t+1}}$ 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 suffices to show that the compilation process represented as the new obfuscator's construction is simulatable. We show a simulator S (with access to Θ) that works as follows: given an obfuscated circuit B in the Θ 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.

Acknowledgements. We thank the anonymous reviewers of Crypto 2017 for their useful comments.

References

1. Prabhanjan Ananth and Abhishek Jain. Indistinguishability obfuscation from compact functional encryption. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part I*, volume 9215 of *LNCS*, pages 308–326. Springer, Heidelberg, August 2015. 2, 4, 13, 17
2. 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. 2, 4, 13
3. 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*, pages 646–658. ACM Press, November 2014. 2
4. Benny Applebaum. Bootstrapping obfuscators via fast pseudorandom functions. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT 2014, Part II*, volume 8874 of *LNCS*, pages 162–172. Springer, Heidelberg, December 2014. 4
5. Benny Applebaum and Zvika Brakerski. Obfuscating circuits via composite-order graded encoding. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015, Part II*, volume 9015 of *LNCS*, pages 528–556. Springer, Heidelberg, March 2015. 2
6. Gilad Asharov and Gil Segev. Limits on the power of indistinguishability obfuscation and functional encryption. In *Foundations of Computer Science (FOCS), 2015 IEEE 56th Annual Symposium on*, pages 191–209. IEEE, 2015. 3, 6, 9, 13

7. Gilad Asharov and Gil Segev. On constructing one-way permutations from indistinguishability obfuscation. In *Theory of Cryptography - 13th International Conference, TCC 2016-A, Proceedings, Part II*, pages 512–541, 2016. 3, 9, 13
8. 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
9. Paul Baecher, Christina Brzuska, and Marc Fischlin. Notions of black-box reductions, revisited. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 296–315. Springer, 2013. 6, 9
10. 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, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 221–238. Springer, Heidelberg, May 2014. 2
11. 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, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 1–18. Springer, Heidelberg, August 2001. 2
12. John Bethencourt, Amit Sahai, and Brent Waters. Ciphertext-policy attribute-based encryption. In *2007 IEEE Symposium on Security and Privacy*, pages 321–334. IEEE Computer Society Press, May 2007. 14
13. Nir Bitansky and Vinod Vaikuntanathan. Indistinguishability obfuscation from functional encryption. In Venkatesan Guruswami, editor, *56th FOCS*, pages 171–190. IEEE Computer Society Press, October 2015. 2, 4, 13, 17
14. Dan Boneh and Mark Zhandry. Multiparty key exchange, efficient traitor tracing, and more from indistinguishability obfuscation. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 480–499. Springer, Heidelberg, August 2014. 3
15. Zvika Brakerski, Christina Brzuska, and Nils Fleischhacker. On statistically secure obfuscation with approximate correctness. Cryptology ePrint Archive, Report 2016/226, 2016. http://eprint.iacr.org/. 6, 10, 11, 12
16. Zvika Brakerski, Jonathan Katz, Gil Segev, and Arkady Yerukhimovich. Limits on the power of zero-knowledge proofs in cryptographic constructions. In *Theory of Cryptography Conference*, pages 559–578. Springer, 2011. 3, 13
17. Zvika Brakerski and Renen Perlman. Lattice-based fully dynamic multi-key FHE with short ciphertexts. Cryptology ePrint Archive, Report 2016/339, 2016. http://eprint.iacr.org/2016/339. 2
18. Zvika Brakerski and Guy N. Rothblum. Virtual black-box obfuscation for all circuits via generic graded encoding. In Yehuda Lindell, editor, *TCC 2014*, volume 8349 of *LNCS*, pages 1–25. Springer, Heidelberg, February 2014. 2
19. Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In Rafail Ostrovsky, editor, *52nd FOCS*, pages 97–106. IEEE Computer Society Press, October 2011. 2
20. Zvika Brakerski and Vinod Vaikuntanathan. Fully homomorphic encryption from ring-LWE and security for key dependent messages. In Phillip Rogaway, editor, *CRYPTO 2011*, volume 6841 of *LNCS*, pages 505–524. Springer, Heidelberg, August 2011. 2
21. Ran Canetti, Yael Tauman Kalai, and Omer Paneth. On obfuscation with random oracles. Cryptology ePrint Archive, Report 2015/048, 2015. http://eprint.iacr.org/. 6, 10, 11, 24
22. 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, Part II*, volume 9015 of *LNCS*, pages 468–497. Springer, Heidelberg, March 2015. 4

23. Jung Hee Cheon, Kyoohyung Han, Changmin Lee, Hansol Ryu, and Damien Stehlé. Cryptanalysis of the multilinear map over the integers. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part I*, volume 9056 of *LNCS*, pages 3–12. Springer, Heidelberg, April 2015. 2
24. Michael Clear and Ciaran McGoldrick. Multi-identity and multi-key leveled FHE from learning with errors. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 630–656. Springer, Heidelberg, August 2015. 2
25. Jean-Sébastien Coron, Craig Gentry, Shai Halevi, Tancrède 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, *CRYPTO 2015, Part I*, volume 9215 of *LNCS*, pages 247–266. Springer, Heidelberg, August 2015. 2
26. Jean-Sébastien Coron, Moon Sung Lee, Tancrède Lepoint, and Mehdi Tibouchi. Cryptanalysis of GGH15 multilinear maps. Cryptology ePrint Archive, Report 2015/1037, 2015. http://eprint.iacr.org/2015/1037. 2
27. Jean-Sébastien Coron, Tancrède Lepoint, and Mehdi Tibouchi. Practical multilinear maps over the integers. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 476–493. Springer, Heidelberg, August 2013. 2
28. Dana Dachman-Soled. Towards non-black-box separations of public key encryption and one way function. In *Theory of Cryptography Conference*, pages 169–191. Springer, 2016. 7
29. Yevgeniy Dodis, Shai Halevi, Ron D. Rothblum, and Daniel Wichs. Spooky encryption and its applications. LNCS, pages 93–122. Springer, Heidelberg, August 2016. 2
30. Sanjam Garg, Craig Gentry, and Shai Halevi. Candidate multilinear maps from ideal lattices. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 1–17. Springer, Heidelberg, May 2013. 2
31. Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *54th FOCS*, pages 40–49. IEEE Computer Society Press, October 2013. 2, 4
32. 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
33. Sanjam Garg, Omkant Pandey, and Akshayaram Srinivasan. Revisiting the cryptographic hardness of finding a nash equilibrium. In *CRYPTO 2016, Part II*, LNCS, pages 579–604. Springer, Heidelberg, August 2016. 3
34. Sanjam Garg, Omkant Pandey, Akshayaram Srinivasan, and Mark Zhandry. Breaking the sub-exponential barrier in obfustopia. Cryptology ePrint Archive, Report 2016/102, 2016. http://eprint.iacr.org/2016/102. 3
35. Craig Gentry. *A fully homomorphic encryption scheme*. PhD thesis, Stanford University, 2009. crypto.stanford.edu/craig. 4, 17
36. Craig Gentry. Fully homomorphic encryption using ideal lattices. In Michael Mitzenmacher, editor, *41st ACM STOC*, pages 169–178. ACM Press, May / June 2009. 2, 3
37. Craig Gentry, Sergey Gorbunov, and Shai Halevi. Graph-induced multilinear maps from lattices. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015, Part II*, volume 9015 of *LNCS*, pages 498–527. Springer, Heidelberg, March 2015. 2
38. Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 75–92. Springer, Heidelberg, August 2013. 2
39. Craig Gentry and Daniel Wichs. Separating succinct non-interactive arguments from all falsifiable assumptions. In Lance Fortnow and Salil P. Vadhan, editors, *STOC*. ACM, 2011. 7

40. Oded Goldreich and Leonid A. Levin. A hard-core predicate for all one-way functions. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing (STOC)*, pages 25–32, 1989. 21
41. Shafi Goldwasser and Guy N. Rothblum. *On Best-Possible Obfuscation*, pages 194–213. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007. 11
42. Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Attribute-based encryption for circuits. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th ACM STOC*, pages 545–554. ACM Press, June 2013. 2
43. Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Predicate encryption for circuits from LWE. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 503–523. Springer, Heidelberg, August 2015. 2
44. Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *ACM CCS 06*, pages 89–98. ACM Press, October / November 2006. Available as Cryptology ePrint Archive Report 2006/309. 14
45. Thomas Holenstein. Complexity theory, 2015. http://www.complexity.ethz.ch/education/Lectures/ComplexityFS15/skript_printable.pdf. 10
46. Yupu Hu and Huiwen Jia. Cryptanalysis of GGH map. Cryptology ePrint Archive, Report 2015/301, 2015. http://eprint.iacr.org/2015/301. 2
47. Russell Impagliazzo and Steven Rudich. Limits on the provable consequences of one-way permutations. In *21st ACM STOC*, pages 44–61. ACM Press, May 1989. 2, 3, 6, 9
48. Huijia Lin. Indistinguishability obfuscation from constant-degree graded encoding schemes. LNCS, pages 28–57. Springer, Heidelberg, 2016. 2, 4
49. Huijia Lin and Vinod Vaikuntanathan. Indistinguishability obfuscation from DDH-like assumptions on constant-degree graded encodings. Cryptology ePrint Archive, Report 2016/795, 2016. http://eprint.iacr.org/2016/795. 2
50. Mohammad Mahmoody, Ameer Mohammed, and Soheil Nematihaji. More on impossibility of virtual black-box obfuscation in idealized models. Cryptology ePrint Archive, Report 2015/632, 2015. http://eprint.iacr.org/. 6, 10
51. Mohammad Mahmoody, Ameer Mohammed, Soheil Nematihaji, Rafael Pass, and abhi shelat. A note on black-box separations for indistinguishability obfuscation. Cryptology ePrint Archive, Report 2016/316, 2016. http://eprint.iacr.org/2016/316. 6, 10, 11
52. Mohammad Mahmoody, Ameer Mohammed, Soheil Nematihaji, Rafael Pass, and Abhi Shelat. *Lower Bounds on Assumptions Behind Indistinguishability Obfuscation*, pages 49–66. Springer Berlin Heidelberg, Berlin, Heidelberg, 2016. 6, 10, 11
53. 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
54. Eric Miles, Amit Sahai, and Mark Zhandry. Annihilation attacks for multilinear maps: Cryptanalysis of indistinguishability obfuscation over GGH13. Cryptology ePrint Archive, Report 2016/147, 2016. http://eprint.iacr.org/2016/147. 2
55. Pratyay Mukherjee and Daniel Wichs. Two round multiparty computation via multi-key FHE. LNCS, pages 735–763. Springer, Heidelberg, 2016. 2
56. Moni Naor. On cryptographic assumptions and challenges. *Lecture Notes in Computer Science*, 2729:96–109, 2003. 7
57. Rafael Pass. Limits of provable security from standard assumptions. In *Proceedings of the forty-third annual ACM symposium on Theory of computing*, pages 109–118. ACM, 2011. 7
58. Rafael Pass and abhi shelat. Impossibility of vbb obfuscation with ideal constant-degree graded encodings. Cryptology ePrint Archive, Report 2015/383, 2015. http://eprint.iacr.org/. 6, 10

59. Rafael Pass, Wei-Lung Dustin Tseng, and Muthuramakrishan Venkitasubramaniam. Towards non-black-box separations in cryptography. TCC, 2011. 7
60. Chris Peikert and Sina Shiehian. Multi-key FHE from LWE, revisited. Cryptology ePrint Archive, Report 2016/196, 2016. http://eprint.iacr.org/2016/196. 2
61. Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *37th ACM STOC*, pages 84–93. ACM Press, May 2005. 2
62. Omer Reingold, Luca Trevisan, and Salil P. Vadhan. Notions of reducibility between cryptographic primitives. In *Theory of Cryptography, First Theory of Cryptography Conference, TCC 2004*, volume 2951 of *Lecture Notes in Computer Science*, pages 1–20. Springer, 2004. 2, 3, 6, 7, 8, 9, 13
63. Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In David B. Shmoys, editor, *46th ACM STOC*, pages 475–484. ACM Press, May / June 2014. 2, 3
64. Amit Sahai and Brent R. Waters. Fuzzy identity-based encryption. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 457–473. Springer, Heidelberg, May 2005. 14
65. Joe Zimmerman. How to obfuscate programs directly. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 439–467. Springer, Heidelberg, April 2015. 2