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

Sanjam Garg* Mohammad Mahmoody† Ameer Mohammed‡

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 Indistinguishability Obfuscation. We show that (assuming that the polynomial hierarchy does not collapse and one-way functions exist) IO cannot be constructed from powerful encryption primitives, such as witness encryption (WE), predicate encryption, and fully homomorphic encryption even using some powerful non-black-box techniques, formally defined as "monolithic" constructions (more details below). 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.

Monolithic Constructions: New Model for Non-Black-Box 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), Reingold, Trevisan, and Vadhan (TCC'04). Our model builds upon and extends the previous non-black-box model of Brakerski, Katz, Segev, and Yerukhimovich (TCC'11) and Asharov and Segev (FOCS'15), by allowing the non-black-box technique of self-feeding a primitive's code to *itself* as oracle calls planted inside input circuits. Since we allow *any* gates to be planted in the input circuits as long as they treat the subroutines of the primitive monolithically, we call such constructions *monolithic* and separations for all such constructions *monolithic separations*.

*University of California, Berkeley. 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.

†University of Virginia. Supported by NSF CAREER award CCF-1350939.

‡University of Kuwait. Work done while at the University of Virginia and visiting UC Berkeley.

Contents

1 Introduction

Program obfuscation provides an extremely powerful tool to make computer programs "unintelligible" while preserving their functionality. Barak, Goldreich, Impagliazzo, Rudich, Sahai, Vadhan and Yang [BGI+01] 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 [GGH+13b] 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" [SW14].

Since the original work of [GGH+13b] many constructions of IO were proposed [GGH+13b, BGK+14, BR14] [AGIS14, MSW14, AB15, Zim15, BMSZ15, GMM+16]. However, all these constructions are based on computational hardness assumptions on multilinear maps [GGH13a, CLT13, GGH15]. Going a step further, recent works of Lin [Lin16] and Lin and Vaikunthanatan [LV16] 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 [AJ15, BV15]. 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 [GGH13a, CHL+15, HJ15, CGH+15, CLLT15, MSZ16].

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 [Reg05]. LWE is already known to imply attribute-based encryption [GVW13] (or even predicate encryption [GVW15]), fully homomorphic encryption [Gen09b, BV11b, BV11a, GSW13][1], multi-key [CM15, MW16, BP16, PS16] and spooky homomorphic encryption [DHRW16a]. 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.[2] 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 [IR89, RTV04] has limited value as we argue below.[3] 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] Realizing full-fledged fully-homomorphic encryption needs additional circular security assumptions.

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

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

[AJ15, BV15, AJS15] is non-black-box in its use of functional encryption. This is not a coincidence (or, just one example) and many applications of functional encryption (as well as other powerful encryption schemes) and IO are non-black-box [Gen09b, SW14, BZ14, GPS16, GPSZ16]. 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 [IR89, RTV04]. In particular, this more general framework should incorporate the aforementioned non-black-box techniques as part of the framework itself.

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

In this work, we revisit the models used in [BKSY11, AS15, AS16] who allowed the use of one-way function gates inside the given circuits and study a model where there is no limitation on what type of oracle gates could be used in the circuits given as input to the special subroutines, and in particular, the primitive's own subroutines could be planted as gates in the input circuits. We believe a model that captures the "gate plantation" technique without putting any limitation on the types of gates used is worth to be studied directly and at an abstract level, due to actual *positive* results that exactly benefit from this "self-feeding" non-black-box technique. For this goal, here we initiate a formal study of a model that we call the *monolithic* model, which captures the above-described non-black-box technique that is commonplace in constructions that use primitives

with subroutines that take arbitrary circuits as input.

More formally, suppose $\mathcal{P}$ 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. A *monolithic* 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 monolithic model are as follows.

- Gentry's bootstrapping construction [Gen09a] plants FHE's own decryption gates inside a circuit that is given as input to the evaluation subroutine. This trick falls into the monolithic framework since planting gates inside evaluation circuits is allowed.
- The bootstrapping of IO for $\mathbf{NC}_1$ (along with FHE) to obtain IO for $\mathbf{P/poly}$ [GGH+13b]. 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 [App14, CLTV15] also fall in our framework.
- The construction of IO from functional encryption [AJ15, BV15, AJS15] plants the functional encryption scheme's encryption subroutine inside the circuits for which decryption keys are issued. Again, such a non-black-box technique does fall into our monolithic framework. We note that the constructions of obfuscation based on constant degree graded encodings [Lin16] also fit in our framework.

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

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

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

In summary, intuitively, our model provides a way to capture "black-box" uses of the known non-black-box techniques. While the full power of non-black-box techniques in cryptography is yet to be understood, virtually every known use of non-black-box techniques follows essentially the same principles, i.e. by 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.

1.1 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 a *monolithic* construction as described above. (A summary of our results is presented in Figure 1.)

Theorem 1.1 (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* **NP** $\not\subseteq$ **coAM***, there is no construction of IO from $\mathcal{P}$ in the monolithic 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 1.1 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 be enough for realizing IO. We expect that our techniques will be useful in deriving impossibility results in such case.

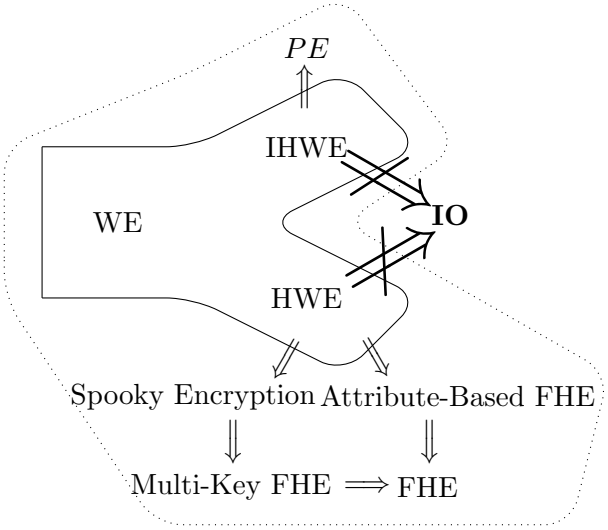

Figure 1: Summary of our results. IHWE denotes Instance Hiding WE and HWE denotes Homomorphic Witness Encryption.

What does our results say about LWE? Even though our separations of Theorem 1.1 covers most of the powerful LWE-based primitives known to date, it does not imply whether or not we can actually base IO on LWE. In fact, our result only rules out specific paths from LWE toward IO that would go through either of the primitives listed in Theorem 1.1. 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 [GGSW13] 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 imply IO in a monolithic way. Next, we show that these two primitives imply extended versions of all the all-or-nothing primitives listed above in a monolithic way. The final separations follow from a specific transitivity lemma that holds in the monolithic model.

1.2 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 [IR89, RTV04].

Previous lower bounds on complexity of IO. The work of Mahmoody et. al [MMN+16b] proved lower bounds on the assumptions that are needed for building IO in a fully black-box way.[4] They showed that, assuming **NP** $\neq$ **co-NP**, one-way functions or even collision resistant hash functions do not imply IO in a fully black-box way.[5] Relying on the works of [CKP15, Pas15, MMN15] (in the context of VBB obfuscation in idealized models) Mahmoody et. al [MMN+16b] 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 [BBF16] that ruled out the existence of statistically secure *approximately correct* IO and a variant of Borel-Cantelli lemma, Mahmoody et. al [MMN+16a] showed how to extend the 'hardness of constructing IO' result of [MMN+16b] 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 [BBF13] studied various generalizations of the black-box framework of [RTV04] that also allow some forms of non-black-box use of primitives. The work of Pass, Venkitasubramaniam and Tseng [PTV11] 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 [Pas11] 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 [Nao03] as long at the security proof is black-box (even if the construction is non-black-box). The work of Gentry and Wichs [GW11] 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 [DS16] showed that certain classes

[4] A previous result of Asharov and Segev [AS15] 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 4 where we formalize the notion of such stronger primitives in a general way.).

[5] Note that since statistically secure IO exists if **P** = **NP**, therefore we need computational assumptions for proving lower bounds for assumptions implying IO.

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. In Section 2, we present at a high-level the main technical ideas behind this work. In Section 3, 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 4, we discuss the monolithic model and its relation to extended primitives in detail and give a formal definition of monolithic constructions from witness encryption. In Section 5, we give a full proof of the separation of IO from (even instance-revealing) extended witness encryption. In Section 6, we give a full proof of the separation of IO from extended instance-hiding witness encryption. In Section 7, we give a full proof of the separation of IO from extended homomorphic witness encryption. Finally, in Section 8, we show how to get different extended primitives from the stronger variants of witness encryption (which we separated from IO in previous sections) in order to conclude that these extended primitives are also separated from IO.

2 Technical Overview

2.1 New Extended Framework for Proving Separations

In this section, we describe ideas that lead to an extension to the black-box framework of [IR89, RTV04], which we call the *monolithic framework* where a common *non-black-box* technique is allowed in the constructions. The goal of this section is to explain and justify the way we define "monolithic" constructions that use specific primitives such as witness encryption. As it will become clear later, our definition of monolithic constructions is meaningful with respect to a "special" primitive with a "special" subroutine that takes a general circuit as input.[6] But, to be fully formal, we will give formal definitions of what this monolithic model means when we use the exact set of primitives mentioned in our main theorem. This way, we can give *full formal definitions as to what we mean by an monolithic construction for our specific primitives* that are needed for the separations in Theorem 1.1, and we also give full proofs of separations for them under this more relaxed black-box notion. However, we emphasize that, for a new primitive $\mathcal{P}$, one still has to give a formal definition of what it means to use $\mathcal{P}$ in an monolithic way by specifying its special subroutines. What we discuss in this section lays down the main intuitive features that, if $\mathcal{P}$ possesses them, a monolithic use of $\mathcal{P}$ is possible.

Intuitively, this monolithic model allows the construction of a primitive $\mathcal{Q}$ based on another primitive $\mathcal{P}$ to plant oracle gates (or oracle calls) to $\mathcal{P}$ itself in the circuits (or Turing machines) in generic computations done by $\mathcal{P}$ whenever $\mathcal{Q}$ has the choice of doing so by choosing certain inputs given to $\mathcal{P}$. For example, to get identification protocols from one-way functions and zero-knowledge proofs [FS86, FFS88] one has to use the code of one-way functions and feed it to the zero-knowledge protocol used. This makes the final construction *non-black-box* according to the [RTV04] definition, however we would like to define a general abstract extended model of black-box constructions (i.e. monolithic constructions) that allows such techniques.

In Section 4 we get into the details of how to define black-box reductions for a class of special primitives with a special subroutine as described above that fall under the monolithic framework. In this technical overview, however, for sake of simplicity and concreteness at the same time, we

[6] This is not the formal definition, as the input circuit should be used in certain ways.

focus on developing a formal definition of what it means to use *witness* encryption in a monolithic way. As we said in the introduction, witness encryption will indeed play a crucial role in deriving our impossibility results.

2.1.1 A Concrete Definition for Case of WE

Consider the primitive of witness encryption [] $(\mathrm{Enc}, \mathrm{Dec})$ where one can run $\mathrm{Enc}(a, m) = c$ to encrypt a message m under a circuit a, and then later on, one can also try to decrypt c by using a "witness" w for which $a(w) = 1$.[7] A black-box construction (see Definition 3.12) of a primitive based on witness encryption is limited to only encrypting messages under circuits a that are in the *plain* model. That is because the input-output function definition of primitive witness encryption (as stated above) only accepts plain circuits as input. However, in many applications of witness encryption, when we use witness encryption, we end up using *oracle* circuits a^O. The reason that doing so is fine (even though the definition of witness encryption requires plain circuits) that those procedures of O would be eventually implemented efficiently and could be turned into circuits (using the Cook-Levin reduction).

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

Monolithic construction = fully black-box use of an extended primitive The above-mentioned example of how a monolithic construction using WE works shows that monolithic constructions using witness encryption could be interpreted in an equivalent way in which the construction is in fact *fully* black-box (see Definition 3.12), but it uses a *stronger* "extended" variant of the WE primitive where we allow oracle gates (calling the same primitive, namely witness encryption) to be planted in the circuits given as input to the encryption subroutine. Note that the extended WE is technically a *different* primitive than WE (because for inefficient implementations of WE, we would accept inefficient circuits as inputs as well), even though the two primitives are equivalent in a *non-black-box* way.

The following is the way we define extended WE in a "symmetric" way such that either the given input instance is a circuit (by interpreting it so and running it over the witness), or the instance is actually an input to a circuit and it is the *witness* that is the circuit. See Definition 4.6 for a more detailed version.

Definition 2.1 (Extended Witness Encryption)**.** Let V be an *oracle* algorithm that takes "instance" a and "witness" w as inputs, interprets a as an oracle-aided circuit, then outputs $a^{(\cdot)}(w)$. An *extended witness encryption* scheme for a relation given by V consists of two PPT algorithms $(\mathrm{Enc}, \mathrm{Dec}_{\mathsf{V}})$ defined as follows:

[7] This way of defining witness encryption is based on the **NP** complete problem of circuit satisfiability.

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

An extended witness encryption scheme satisfies the the same completeness and security properties as the standard Definition of WE (see Definition 3.3).

We can also define a symmetric variant of extended WE where the relation V interprets w as the circuit instead and runs $w(a)$. In either case, we note that, since V is a relation that acts as a universal circuit, the running time of $\mathsf{V}(w, a)$ is guaranteed to be $\mathrm{poly}(n)$ where $n = |a| + |w|$. That is due to the fact that each recursive call is invoked on inputs of a smaller size (in fact of length $n - 1$) which results in a halting execution of time at most $O(n^2)$.

Using the same idea, one can define extended variants of many other primitives (see Definition 4.7). Finally, after defining what an extended WE primitive is, we can formally define what it means to have an monolithic construction that uses WE:

Definition 2.2 (Monolithic Construction of WE)**.** Suppose $\mathcal{Q}$ is a primitive and $\widetilde{\mathsf{WE}}$ is an extended version of WE defined as in Definition 2.1. Any fully black-box construction (see Definition 3.12) for $\mathcal{Q}$ from $\widetilde{\mathsf{WE}}$ (i.e. an extended version of WE) is also a *monolithic* construction of $\mathcal{Q}$ from WE.

Again, the idea extends beyond WE and we apply it to all the other primitives that we deal with in our main Theorem 1.1.

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

2.1.2 A Transitivity Lemma for Deriving More Separations

The way we defined monolithic constructions (based on black-box constructions using the extended primitive) allows us to prove a transitivity lemmas that will then pave our way towards proving more separations in the monolithic framework. Namely, it can be shown that if $\mathcal{P}, \mathcal{Q}, \mathcal{R}$ are cryptographic primitives and (1) there is an monolithic construction of the *extended* variant of $\mathcal{Q}$ from $\mathcal{P}$ and (2) there is an monolithic construction of $\mathcal{R}$ from $\mathcal{Q}$, then there is a monolithic construction of $\mathcal{R}$ from $\mathcal{P}$ (see Lemma 4.10 and its proof). Therefore, to prove that a primitive $\mathcal{P}$ (e.g., predicate encryption) does not imply IO, we could employ the following argument outline:

1. Separate IO from WE in the monolithic model. In other words, prove that there is no fully black-box construction of IO from the extended variant of WE.
2. Prove that the *extended* variant of the primitive $\mathcal{P}$ could be obtained from witness encryption in a monolithic way.

We will apply the above idea to various forms of witness encryption and different all-or-nothing encryption primitives $\mathcal{P}$.

2.2 Using Variants of Witness Encryption as the Middle Primitive

As described above, to derive more separation lower bounds for IO, e.g., from a primitive $\mathcal{Q}$, we could define a "middle primitive" $\mathcal{P}$ and the following: (1) show that $\mathcal{P}$ does not imply IO in the monolithic model, and (2) show that there is a monolithic construction of the *extended* version of $\mathcal{Q}$ from $\mathcal{P}$. In this work we will use variants of WE to play the role of this middle primitive $\mathcal{P}$. Namely we will use the following two variants: (1) "instance-hiding" WE, and (2) "instance-revealing homomorphic" WE.[8] Below we will discuss these notions in more details, but before doing that we shall point out that these two notions are *incomparable* strengthenings of the basic notion of witness encryption. The first one strengthens witness encryption in terms of security (which is essential for achieving predicate encryption) while the latter strengthening is in terms of functionality (which is essential for achieving full-homomorphic encryption and its variants). These two notions are incomparable and in fact strengthen the standard notion of witness encryption in somewhat incomparable ways. As a side remark, we note that strengthening witness encryption to incorporate instance hiding property precludes us from also having compact homomorphism along with. Therefore, we need to consider these two primitives separately. Furthermore, showing that obfuscation is not possible from either of them poses different technical challenges.

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

2.2.1 Known Recipe for Proving Lower Bounds for IO

A sequence of recent work [CKP15, MMN15, Pas15, MMN+16b, BBF16, MMN+16a] has led to the following (informally stated) lemma which provides a way to prove fully lower bounds for IO.

Lemma 2.3 (Lemma for Proving Lower Bounds for IO)**.** *Suppose $\mathcal{I}$ is an idealized oracle and suppose we can "compile out" $\mathcal{I}$ from any IO construction in a world where $\mathcal{I}$ is accessible to get an IO scheme in the plain model where the new scheme is only approximately correct: it computes the correct answer only over $99/100$ fraction of the input. Also suppose primitive $\mathcal{P}$ can be securely implemented in the model $\mathcal{I}$. Then there is no fully black-box construction of IO from $\mathcal{P}$.*

See Section 3.4 (particularly Lemma 3.26 there) for details of how to derive this lemma from previous works. Below we discuss how we apply this approach to the case of variants of witness encryption.

2.2.2 The Warm-Up Case: Applying Lemma 2.3 to the Basic Case of WE

To separate IO from the *extended* variant of WE (i.e., separating IO from WE in the monolithic model) we can try to apply Lemma 2.3 and prove a *fully* black-box separation for IO from the

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

extended variant of WE. Suppose V is the oracle universal circuit algorithm that defines the witness verification of the *extended* WE primitive we would like to separate from IO. As a first try, let see what happens if we try to use the following simple oracle to implement this extended WE primitive.

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

1. If $\nexists\, x$ such that $\mathrm{Enc}(x) = c$, output $\perp$. Otherwise:
2. Find x such that $\mathrm{Enc}(x) = c$ and parse it as $x = (a, m)$.
3. If $\mathsf{V}^{\Theta}(w, a) = 0$ output $\perp$. (Note that this step might need launching some recursive oracle calls.) Otherwise, output m.

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

Trying to compile out $\mathcal{I}$ from any IO scheme $\mathrm{IO}^{\mathcal{I}}$. Let $\mathrm{IO}^{\mathcal{I}}$ be an obfuscation in some idealized model $\mathcal{I}$ that implements the extended WE primitive. As mentioned above, one of our main tasks is to "compile-out" the oracle from an ideal model secure obfuscator to get an approximate plain-model obfuscator IO' that is also secure. To do so we use the ideas developed in the work of [CKP15, MMN15, Pas15] which is based on learning the "heavy queries" asked by the obfuscated code to the oracle and hard-coding them into the obfuscated code in the plain model.[9] Specifically, the new plain-model obfuscator IO', given a circuit C to obfuscate would work in two steps. The first step of IO' is to emulate $\mathrm{IO}^{\mathcal{I}}(C)$ to get an ideal-model obfuscation B, making sure to lazily evaluate (i.e., emulate) any queries issued to $\mathcal{I}$. The second step of IO' is to learn the queries that are "likely" to be asked by $B^{\mathcal{I}}(x)$ for a random input x. We do this by executing $B^{\mathcal{I}}(x_i)$ (while we emulate the queries to $\mathcal{I}$ consistently) enough number of times for different x_i in an effort to learn all the highly probable queries, which we denote by Q_B. The output of IO' is the plain-model obfuscation $B' = (B, Q_B)$, where B is the ideal-model obfuscation and Q_B is the set of learned queries. To evaluate the obfuscation over a new random input x, we simply execute $B'(x) = B^{\mathcal{I}}(x)$ while emulating any queries to $\mathcal{I}$ consistently relative to Q_B.

Security of compiler: only include simulatable information in B'. The security of the new plain-model obfuscator crucially depends on whether the queries Q_B that we publish are simulatable. In other words, we want to prove that if the adversary A against this new plain-model does not gain any additional advantage because these queries can be simulated by having A itself run $B^{\mathcal{I}}(x_i)$ several times. As a result, we will *only* put queries in Q_B (forwarded to be

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

part of the plain-model obfuscator) where these queries could be obtained by the adversary as well. Doing so allows us to immediately reduce the security to that of the ideal-model obfuscation. It is therefore important to release only those queries that can be simulated to ensure security is preserved (for example, we cannot publish the queries asked during the emulation in step 1 as they are not simulatable).

The other task we will have is to prove the *approximate* correctness of the new model, which informally states that an execution $B'(x)$ should be correct with high probability over x. Note that unless we ask an unlearned query to $\mathcal{I}$ that was previously asked by some hidden part of the emulation process (e.g. during step 1), the execution should be statistically distributed to an ideal execution of $B^{\mathcal{I}}(x)$ where the queries are not emulated but answered using a real oracle. In fact if the oracle $\mathcal{I}$ did not involve any "internal/hidden" query asked by V, we could use the arguments given [CKP15, MMN15] to show that: the probability that we ask an unlearned hidden query occurs with sufficiently small probability.

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

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

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

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

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

2.3 Separating IO from Instance-Hiding WE

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

is strong enough to imply (even extended) PE in a monolithic way.[10] An instance hiding WE is a variant of WE where two ciphertexts $\mathrm{Enc}(a_0, m_0) = c_0, \mathrm{Enc}(a_1, m_1) = c_1$ that hide *different* instance/attributes $a_0 \neq a_1$ are indistinguishable so long as there is no witness that satisfies either of these instances; namely $\mathsf{V}(a_b, w) = 0$ for all w and $b \in \{0, 1\}$ (see Definition 3.5). The reason that we will need such variant of WE for constructing PE is that PE itself has the same nature and different ciphertexts under unsatisfiable attributes should remain indistinguishable.

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

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

1. If $\nexists\, x$ such that $\mathrm{Enc}(x) = c$, output $\perp$. Otherwise:
2. Find x such that $\mathrm{Enc}(x) = c$ and parse it as $x = (a, m)$.
3. If $\mathsf{V}^{\Theta}(w, a) = 0$ output $\perp$. (Note that this step might launch some recursive oracle calls.) Otherwise, output *both* of (a, m).

Note that the above modified way of decryption ciphertexts will *not* interfere with the instance hiding property of the scheme. However, it will be hugely important for us to still be able to discover the heavy queries of the obfuscated code in the idealized model (and so we can still compile out the idealized oracle from any IO construction). Unfortunately, we are not able to discover all the hidden/internal queries of a decryption query immediately, but we can do so only conditionally. Specifically, whenever the obfuscated circuit calls a decryption query $\mathrm{Dec}(w, c)$ and the underlying message x is successfully decrypted, the obfuscated circuit can now run $\mathsf{V}^{\Theta}(w, a) = 1$ to reveal the queries asked. This can be thought of as a weaker form of canonicalization of the code of the verifier that still *sometimes* reveals the hidden parts of the oracle computation. That is because, it is possible that the decryption fails, in which case we may have some hidden queries Q_{V} asked by an underlying $\mathsf{V}^{\Theta}(w, a) = 0$ that cannot be revealed or learned. Nonetheless, we show using a careful argument that by doing enough rounds of learning (over random inputs) and "opening up" the internal hidden parts of the verification queries as much as possible, we will still get enough information to run the final (plain model) obfuscated code B' with approximate correctness close to 1. We refer the reader to the next sections for formal proofs.

2.4 Separating IO from Homomorphic WE

For obtaining our separations for IO from the set of "homomorphic" primitives listed in Theorem 1.1 we will employ a "middle" primitive called "homomorphic (instance revealing) witness encryption".

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

Roughly speaking, this primitive allows homomorphic operations over the ciphertexts that are encrypted under the *same* attribute/instance. However, it is possible to always extract this attribute efficiently (this is exactly how we modified the witness encryption in our basic case to be able to prove the separation from IO). In other words, we will need an extra functionality over the instance revealing WE as described in previous section.

New challenge: homomorphic queries. Due to the fact that we are still working with an *instance revealing* form of WE, this simplifies our job and we will *not* fall into the challenges that we faced for the case of Instance-hiding WE. However, we will have a new challenge: we need to find the heavy queries that are asked during the *evaluation* procedure! This is something that did not exist in the basic and instance hiding versions of WE. In Section 7 we will show that, essentially, the same learning procedure will allow us to learn enough information to simulate the last final (real) execution of the obfuscated code on a given input. In particular, for an evaluation query $\mathsf{Eval}_{\mathsf{F}}(c_1, \ldots, c_k)$ that computes some homomorphic evaluation $\mathsf{F}(m_1, \ldots, m_k)$ over the plaintexts inside $(c_1, \ldots, c_k)$, there are two cases: either we already have learned the messages $m_1, \ldots, m_k$ in which case we can also run the algorithm $\mathsf{F}(m_1, \ldots, m_k)$, but we do not know *any* of these messages, we will simply generate a random answer and use it to emulate the answer to the homomorphic operation. A careful analysis is needed to prove that this in fact leads to an approximately correct execution of the code in the plain model.

2.5 Primitives Implied by Our Variants of WE

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

2.5.1 Getting Extended PE from Instance Hiding WE

We start by showing how instance-hiding WE implies extended selectively-secure PE for the predicate $\mathsf{P}(k, a)$ where $k \in K$ and $a \in A$.[11] The idea is straightforward. A secret key for a string k is just a signature on it and a ciphertext encrypting a message $x = (a, m)$ is an (instance hiding) witness encryption with respect to the attribute C that represents a Boolean circuit for verifying the signature and checking that $P(k, a) = 1$.

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

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

2.5.2 Getting Extended FHE from Homomorphic WE

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

3 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 **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 **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.

3.1 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 [RTV04] 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 3.12).

Definition 3.1 (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 of the same size and compute the same function we have:

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

Definition 3.2 (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:

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

Definition 3.3 (Witness Encryption (WE) Indexed by Verifier V)**.** Let L be an **NP** language with a corresponding efficient relation verifier V (that takes instance a and witness w and either accepts or rejects). A *witness encryption* scheme for relation defined by V consists of two PPT algorithms $(\text{Enc}, \text{Dec}_\mathsf{V})$ defined as follows:

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

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

- **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:

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

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 universal circuit algorithm and $\mathsf{V}(w, a) = 1$ if $a(w) = 1$ where a is a circuit evaluated on witness w.

We also define variants of witness encryption, which are both a strengthening of the Definition 3.3, one of them strengthens WE's functionality and the other one strengthens its security. Therefore these variants of WE are incompatible in their capabilities and, hence, a witness encryption scheme may possess either one or the other extra feature (but not both).

Definition 3.4 (Instance-Revealing Witness Encryption (IRWE))**.** A witness encryption scheme is said to be *instance-revealing* if it satisfies the properties of Definition 3.3 and, in addition, includes the following subroutine.

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

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

Definition 3.5 (Instance-hiding Witness Encryption (IHWE))**.** A witness encryption scheme is said to be *instance-hiding* if it satisfies the properties of Definition 3.3 except that the security property is replaced with the following (stronger) security guarantee:

- **Instance-hiding security.** For any PPT adversary A, there exists a negligible function $\mu(.)$ such that for all $a_0, a_1 \notin \mathsf{V}$ for which $|a_0| = |a_1|$ and m_0, m_1 for which $|m_0| = |m_1|$, the following holds:

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

Definition 3.6 (Predicate Encryption (PE) [BSW10])**.** Let $P_{K,A} : K \times A \to \{0,1\}$ be an efficiently computable predicate defined over some key space K (that contains a special empty key ϵ) and attribute space A and P is an efficient Turing machine computing the relation $P_{K,A}$. A *predicate encryption* scheme for a class of predicates $P_{K,A}$ consists of four PPT algorithms $(\mathrm{Setup}, \mathrm{KGen}, \mathrm{Enc}, \mathrm{Dec}_\mathsf{P})$ defined as follows:

- $\mathrm{Setup}(1^\kappa)$: given the security parameter, it outputs a master public key MPK and a master secret key MSK.
- $\mathrm{KGen}(\mathsf{MSK}, k)$: given $k \in K$ and the master secret key $\mathsf{MSK} \in \{0,1\}^n$, outputs the decryption key sk_k. If $k = \epsilon$, it outputs ϵ.
- $\mathrm{Enc}(\mathsf{MPK}, (m, a))$: given the master public key MPK, attribute $a \in A$, and message m, outputs ciphertext c.
- $\mathrm{Dec}_\mathsf{P}(sk_k, c)$: given a secret key for $k \in K$ and a ciphertext c, outputs a string m (or $\perp$).

The following completeness and security properties must be satisfied:

- **Completeness.** For any security parameter κ, key $k \in K$, attribute $a \in A$, we have:

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

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

- **Security.** For any PPT adversary A, there exists a negligible function $\mathrm{negl}(.)$ such that the following holds:

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

  where $\mathrm{IND}_A^{\mathrm{PE}}$ is shown in Figure 2, and for each key query k that A sends to the KGen oracle, it must hold that $\mathsf{P}(k, a_0) = \mathsf{P}(k, a_1) = 0$.

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

Figure 2: The IND_A^{PE} Experiment

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

We also present the definition of attribute-based encryption, which is a special case of predicate encryption where the attribute of the encrypted message could be potentially extracted efficiently from the ciphertext[12].

Definition 3.7 (Attribute-Based Encryption (ABE))**.** An *attribute-based encryption* scheme is a predicate encryption scheme for a class of predicates $P_{K,A}$ (defined through efficient test $\mathsf{P}(k, a) \in \{0,1\}$) satisfying the properties of Definition 3.6 except that the completeness and security conditions are replaced with the following:

- **Completeness.** For any security parameter κ, key $k \in K$, attribute $a \in A$, and message $m \in M$, the following holds:

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

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

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

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

  where $\text{IND}_A^{\text{ABE}}$ is shown in Figure 3, and for each key query k that A sends to the KGen oracle, it must hold that $P(k, a) = 0$.

Experiment $\text{IND}_A^{\text{ABE}}(1^\kappa)$:
1. $(\mathsf{MSK}, \mathsf{MPK}) \leftarrow \text{Setup}(1^\kappa))$
2. $(x_0, x_1) \leftarrow A^{\text{KGen}(\mathsf{MSK},.)}(\mathsf{MPK})$ where $x_b = (a, m_b)$ for $b \in \{0,1\}$, $|m_0| = |m_1|$ and for each prior query k we require that $P(k, a) = 0$
3. $b \xleftarrow{\$} \{0,1\}$

[12]The work of [BSW10] refers to this sub-class as predicate encryption with public index

4. $c \leftarrow \text{Enc}(\mathsf{MPK}, x_b)$
5. $b' \leftarrow A^{\text{KGen}(\mathsf{MSK},.)}(\mathsf{MPK}, c)$ where for each query k we require that $P(k, a) = 0$
6. Output 1 if $b = b'$ and 0 otherwise.

Figure 3: The $\text{IND}_A^{\text{ABE}}$ Experiment

Definition 3.8 (Fully Homomorphic Encryption (FHE))**.** Let F be a PPT algorithm that accepts as input string f and messages $m_1, ..., m_t$ and outputs a string m'. For any security parameter κ, a *homomorphic scheme* HE for F is composed of four PPT algorithms $(\text{Setup}, \text{Enc}, \text{Dec}, \text{Eval}_{\mathsf{F}})$ defined as follows:

- $\text{Setup}(1^\kappa)$: given the security parameter κ, outputs the master public key MPK and the master secret key MSK.
- $\text{Enc}(\mathsf{MPK}, m)$: given MPK and a message $m \in \{0, 1\}$, outputs an encryption c.
- $\text{Eval}_{\mathsf{F}}(\mathsf{MPK}, f, c_1, ..., c_t)$: given master public key MPK, string f, and a sequence of ciphertexts $c_1 = \text{Enc}(\mathsf{MPK}, m_1), ..., c_t = \text{Enc}(\mathsf{MPK}, m_t)$, outputs another ciphertext c_f.
- $\text{Dec}(\mathsf{MSK}, c)$: given MSK and ciphertext c, outputs a bit $m \in \{0, 1\}$.

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

- **Completeness.** For any security parameter κ, string f and messages $m_1, \ldots, m_t \in \{0, 1\}$, it holds that:
$$\Pr[\text{Dec}(\mathsf{MSK}, \text{Eval}_{\mathsf{F}}(\mathsf{MPK}, f, c_1, ..., c_t)) = \mathsf{F}(f, m_1, ..., m_t)] = 1.$$
where $(\mathsf{MSK}, \mathsf{MPK}) \leftarrow \text{Setup}(1^\kappa)$ and $c_i = \text{Enc}(\mathsf{MPK}, m_i)$ for $i \in [t]$.
- **Compactness.** There exists a fixed polynomial $p(.)$ such that, for every string f, the size of the evaluated ciphertexts $|\text{Eval}_{\mathsf{F}}(\mathsf{MPK}, f, c_1, \ldots, c_t)|$ is at most $p(\kappa)$ where, for all $i \in [t]$, $c_i \leftarrow \text{Enc}(\mathsf{MPK}, m_i)$ for some $m_i \in \{0, 1\}$.
- **Security.** For any PPT adversary A, there exists a negligible function $\mu(.)$ such that the following holds:
$$|\Pr[A(\text{Enc}(\mathsf{MPK}, 0)) = 1] - \Pr[A(\text{Enc}(\mathsf{MPK}, 1)) = 1]| \leq \mu(\kappa).$$
where $(\mathsf{MSK}, \mathsf{MPK}) \leftarrow \text{Setup}(1^\kappa)$ and the probability is over the randomness of A and Enc.

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

Definition 3.9 (Multi-key FHE (MFHE) [LATV12, MW16])**.** A multi-key FHE scheme is an FHE scheme where the evaluation sub-routine is replaced with the following:

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

Furthermore, the following properties must be satisfied:

- **Completeness.** For any security parameter κ, messages $m_i \in \{0,1\}$ and function f, we have:
$$\Pr[\mathrm{Dec}(\overrightarrow{\mathsf{MSK}}, \mathrm{Eval}(\overrightarrow{\mathsf{MPK}}, f, c_1, ..., c_t)) = f(m_1, ..., m_t)] = 1$$
where $\overrightarrow{\mathsf{MSK}} = (\mathsf{MSK}_1, ..., \mathsf{MSK}_L)$, $\overrightarrow{\mathsf{MPK}} = (\mathsf{MPK}_1, ..., \mathsf{MPK}_L)$, and for all $j \in [L]$, $(\mathsf{MSK}_j, \mathsf{MPK}_j) \leftarrow \mathrm{Setup}(1^\kappa)$ and for all $i \in [t]$ there exists $j \in [L]$ such that $c_i \leftarrow \mathrm{Enc}(\mathsf{MPK}_j, m_i)$.

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

- **Security.** For any PPT adversary A, there exists a negligible function $\mu(.)$ such that the following holds:
$$|\Pr[A(\mathrm{Enc}(\mathsf{MPK}, 0)) = 1] - \Pr[A(\mathrm{Enc}(\mathsf{MPK}, 1)) = 1]| \leq \mu(\kappa)$$
where $(\mathsf{MSK}, \mathsf{MPK}) \leftarrow \mathrm{Setup}(1^\kappa)$ and the probability is over the randomness of A and Enc.

Definition 3.10 (Spooky Encryption [DHRW16b])**.** Let F be a PPT algorithm that accepts as input a string f and messages $m_1, ..., m_t$ and outputs a sequence of messages $m'_1, ..., m'_t$. A F-*spooky encryption* scheme is an HE scheme for (possibly randomized) F where evaluation is replaced with the following:

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

Furthermore, the following properties must be satisfied:

- **Completeness.** For any security parameter κ, string f and messages $(m_1, ..., m_t)$ where $m_i \in \{0,1\}^{\mathrm{poly}(\kappa)}$ for all $i \in [t]$, it holds that:
$$\{(\mathrm{Dec}(\mathsf{MSK}_1, c'_1), ..., \mathrm{Dec}(\mathsf{MSK}_t, c'_t))\}_\kappa \approx_c \{\mathsf{F}(f, m_1, ..., m_t)\}_\kappa$$
where for all $i \in [t], (\mathsf{MSK}_i, \mathsf{MPK}_i) \leftarrow \mathrm{Setup}(1^\kappa)$, $c_i \leftarrow \mathrm{Enc}(\mathsf{MPK}, m_i)$, and the evaluated ciphertexts are given as $c'_i \leftarrow \mathrm{Eval}_{\mathsf{F}}(f, (\mathsf{MPK}_1, c_1), ..., (\mathsf{MPK}_t, c_t))$.

- **Security.** For any PPT adversary A, there exists a negligible function $\mu(.)$ such that the following holds:
$$|\Pr[A(\mathrm{Enc}(\mathsf{MPK}, 0)) = 1] - \Pr[A(\mathrm{Enc}(\mathsf{MPK}, 1)) = 1]| \leq \mu(\kappa)$$
where $(\mathsf{MSK}, \mathsf{MPK}) \leftarrow \mathrm{Setup}(1^\kappa)$ and the probability is over the randomness of A and Enc.

In [DHRW16b], a special case of spooky encryption called additive function sharing (AFS) spooky encryption was defined, which is essentially F-spooky encryption where F accepts and executes a circuit f_h that outputs a random n-out-of-n additive secret share of $h(m_1, ..., m_t)$. Note that F-spooky encryption supporting the class of all polynomially-sized circuits is a generalization of multi-key FHE [LATV12, MW16]. In particular, if we allowed repetition of public keys in the evaluation algorithm of the MFHE scheme, then we can define the evaluation algorithm of MFHE to simply use the evaluation of the spooky encryption scheme and output $c_f = (c'_1, ..., c'_t)$.

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

- Witness encryption: The relation R_U with corresponding verifier V is defined so that for any input pair (w, a), $\mathsf{V}(w, a) = a(w)$. Thus, this relation gives rise to the language $L_{\mathsf{V}} = \{a \mid \exists w \colon a(w) = 1\}$, which is essentially the language of circuit satisfiability.
- Predicate and attribute-based encryption: The universal predicate class $P_{K,A}$ with corresponding P is defined so that for any $k \in K$ and $a \in A$, $\mathsf{P}(k, a) = k(a)$. Thus, this relation gives rise to the language $L_{\mathsf{P}} = \{k \mid \exists a \colon k(a) = 1\}$.
- (Multi-key) FHE and spooky encryption: The algorithm F representing the scheme is defined so that for any input sequence $(f, m_1, ..., m_t)$, $\mathsf{F}(f, m_1, ..., m_t) = f(m_1, ..., m_t)$.

Given a primitive defined with universal relation R_U with associated verifier V, we can construct the same primitive for any other arbitrary efficiently computable relation R' with associated verifier V'. This is achieved by defining an instance a to be verified by V as a circuit $a := \mathsf{V}'(., a')$ where a' is an instance to be verified by V'. In that case, we have that, for any (w, a'), $\mathsf{V}'(w, a') = 1$ if and only if $\mathsf{V}(w, a) = 1$ since $\mathsf{V}(w, a) = a(w) = \mathsf{V}'(w, a')$.

We can also define a symmetric variant of the universal relation. For example, for PE or ABE, we can let $\mathsf{P}(k, a) = a(k)$ instead[13]. In either case, we will explicitly mention which input is to act as the circuit when we make use of these relations.

3.2 Black-Box Constructions and Separations

Impagliazzo and Rudich [IR89] were the first to formally study the power of "black-box" constructions that relativize to any oracle. Their notion was further explored in detail by Reingold, Trevisan, and Vadhan [RTV04]. The work of Baecher, Brzuska, and Fischlin [BBF13] further studied the black-box framework and 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 3.11 (Cryptographic Primitives [RTV04])**.** 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 3.12 (Black-Box Constructions [RTV04])**.** 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 $\mathcal{Q}$.

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

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 3.13 (Black-box Constructions of IO)**.** A fully-black-box construction of IO from any primitive $\mathcal{P}$ could be defined by combining Definitions 3.12 and 3.1.

By considering the role of the security parameter we can distinguish between attacks that succeed over an "infinitely often" vs. "almost everywhere" security parameters. Therefore, we will work with a more refined definition of cryptographic primitives (and constructions) where the parties also receive a security parameter κ as input. Thus, any function P implementing a primitive $\mathcal{P}$ will take as input a "security parameter" κ and (according to the standard definition of security in cryptography) any adversary A who successfully breaks P would have to "win" over an *infinite* number of security parameters for a "noticeable" advantage (e.g. only winning over security parameters that are powers of two will suffice to call it a successful attack). This subtle issue will be important for us to derive lower bounds on the complexity of IO as our attacks (as part of the separation proofs) only succeed with a constant probability, and that only allows us to derive attacks that succeed over an infinite number of security parameters (see Section 3.4).

The issue of oracles gates. Note that, in any such construction of Definition 3.13, 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 4 we will see that one can also define an *extended* variant of the IO primitive (as it was done in [AS15, AS16]) in which the input circuits have oracle gates.

3.3 Measure Theoretic Tools

By a *probability space* we mean a *measure space* with total measure equal to one, and by $\Pr[E]$ we denote the measure of E. For a sequence of measurable sets $\mathcal{E} = (E_1, E_2, \dots)$, the limit supremum of $\mathcal{E}$ is defined as $\limSup(\mathcal{E}) = \bigcap_{n=1}^{\infty} \bigcup_{m=n}^{\infty} E_m$.

Lemma 3.14 (Borel–Cantelli [Bor09, Can17])**.** *Let $\mathcal{E} = (E_1, E_2, \dots)$ be a sequence of measurable sets over some probability space, and $\sum_{n=1}^{\infty} \Pr[E_i] = O(1)$. Then $\limSup(\mathcal{E})$ has measure zero.*

The following lemma (also used in [MMN+16a]) follows from Exercise 2 of Section 7.3 of [GS01].

Lemma 3.15. *If $\mathcal{E} = (E_1, E_2, \dots)$ is a sequence of measurable sets over some probability space, and $\Pr[E_i] \geq \delta$ for all $i \in \mathbb{N}$, then $\Pr[\limSup(\mathcal{E})] \geq \delta$.*

3.4 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 [CKP15, MMN15, Pas15, MMN+16b, BBF16, MMN+16a] 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 monolithic model. We will indeed use those tools/lemmas by relating the monolithic 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.[14]

Definition 3.16 (Oracle-Fixed Constructions in Idealized Models [MMN+16b])**.** We say a primitive $\mathcal{P}$ has an *oracle-fixed* construction in idealized model $\mathcal{I}$ if there is an oracle-aided algorithm P where:

- **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 $I \xleftarrow{\$} \mathcal{I}$, it holds that A does *not* break P^I.[15]

Remark 3.17. Definition 3.16 is different from saying that $\mathcal{P}$ would exists in a relativized world $I \leftarrow \mathcal{I}$ with measure one. If the security condition was defined with respect to *efficient* attackers, then we could use the Borel–Cantelli lemma to do a union bound over all such attackers, but the number of *bounded-query* attacking algorithms is not countable. Working with Definition 3.16 allows us to still use idealized model $\mathcal{I}$ for the purpose of proving black-box separations (where we want to say $\mathcal{P}$ is not enough to obtain another primitive $\mathcal{Q}$). Definition 3.16 is called "oracle-fixed" in contrast to another "oracle-mixed" definition in which the same security condition holds (with measure one) over the randomness of the $\mathcal{I}$ and A at the same time.

Definition 3.18 (Oracle-Mixed Constructions in Idealized Models [MMN+16a])**.** We say a primitive $\mathcal{P}$ has an *oracle-mixed* black-box construction in idealized model $\mathcal{I}$ if there is an oracle-aided algorithm P such that:

- **Oracle-mixed completeness.** P^I implements $\mathcal{P}$ correctly where the probabilities are also over $I \leftarrow \mathcal{I}$.[16] 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 $\mathcal{I}$.

[14] 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 [Hol15] for a proof.

[15] 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.

[16] 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.

Lemma 3.19 (The Composition Lemma [MMN+16b])**.** *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 3.16). Then Q^P is an oracle-fixed implementation of $\mathcal{Q}$ relative to the same idealized model $\mathcal{I}$.*

Using a variant of the Borel–Cantelli lemma, [MMN+16a] proved that oracle-mixed attacks with *constant* advantage leads to breaking oracle-fixed constructions.

Lemma 3.20 ([MMN+16a])**.** *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 3.19 and 3.20.

Lemma 3.21 (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 $\mathcal{P}$.

3.5 Tools for Getting Black-Box Lower Bounds for IO

The specific techniques for proving separations for IO that is developed in [CKP15, MMN+16b, BBF16, MMN+16a] aims at employing Lemma 3.21 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 [GR07] 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 [BBF16] who ruled out the existence of statistically secure approximate IO.

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

Definition 3.22 (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 3.23 (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 $\mu(\cdot)$ such that for any C:

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

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

It is easy to see that the existence of the simulator according to Definition 3.23 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 [BBF16] one can get a general method of breaking IO in idealized models. We first state the result of [BBF16].

Theorem 3.24 ([BBF16])**.** *Suppose one-way functions exist,* **NP** $\not\subseteq$ **coAM**, *and* $\delta, \epsilon \colon \mathbb{N} \mapsto [0, 1]$ *are such that* $2\epsilon(n) + 3\delta(n) < 1 - 1/\operatorname{poly}(n)$*, 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 3.25 (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,* **NP** $\not\subseteq$ **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 3.24 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 3.23 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 3.25 and 3.21 together we get a lemma for proving black-box lower bounds for IO.

Lemma 3.26 (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 3.26 to derive lower bounds for IO even in the *monolithic* model by relating such constructions to fully black-box constructions.

4 Monolithic Constructions: An Abstract Extension of the Black-Box Model

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

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 $c = \text{Eval}(f, c_1, \ldots, c_k)$ 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 [SW05, GPSW06]: Here the circuit C is given as input to the key generation algorithm and then $C(m)$ is computed over plaintext m during the decryption.
  - Ciphertext policy [BSW07]: Here the circuit C is the actual plaintext and the input m to C is used when issuing the decryption keys.

  Both of these approaches lead to special primitives. For example, for the case of predicate encryption, suppose we use a predicate verification algorithm P that takes (k, a), interprets k as circuit and runs $k(a)$ to accept or reject. Such P would give us the key policy predicate encryption. Another P algorithm would interpret a as a circuit and runs it on k, and this gives us the ciphertext policy predicate encryption. In other words, one can think of the circuit C equivalent to $\mathsf{P}(k, \cdot)$ (with k hard coded in it, and a left out as the input) being the "input" circuit given to the KGen subroutine, or alternatively one can think of $\mathsf{P}(\cdot, a)$ (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).

4.1 Monolithic Extensions of Special Primitives

We define special primitives as 'restrictions' of (a family of) primitives indexed by a subroutine W to the case that W is a universal circuit evaluator. We then 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 4.1 (Family of Indexed Primitives)**.** Let $\mathcal{W}$ be a *set* of (possibly inefficient) functions. A $\mathcal{W}$*-indexed* family of primitives $\mathcal{P}[\mathcal{W}]$ is a *set* of primitives $\{\mathcal{P}[W]\}_{W\in\mathcal{W}}$ each indexed by some $W \in \mathcal{W}$ where, for each $W \in \mathcal{W}$, $\mathcal{P}[W] = (\mathcal{F}[W], \mathcal{R}[W])$ is a primitive according to Definition 3.11. We call the primitive $\mathcal{P}[W]$ the W-indexed member of the family $\mathcal{P}[\mathcal{W}]$.

We are particularly interested in primitives indexed by the universal circuit evaluation algorithm (defined below) as a member of an indexed family of primitives. This is the case for all the primitives of witness encryption, predicate encryption,[17] fully homomorphic encryption, and IO. All of the examples of the special primitives discussed in previous section fall into this category.

Definition 4.2 (Universal Circuit Evaluator)**.** By oracle-aided algorithm $\mathsf{Univ}^{(\cdot)}$ we denote the *oracle-aided universal circuit evaluator* which accepts a pair of inputs (C, x) where C is an *oracle-aided* circuit and x is a string in the domain of C, and $\mathsf{Univ}(C, x)$ outputs $C^{(\cdot)}(x)$ by forwarding all of C's oracle queries to its own oracle. By algorithm Univ we simply denote the *plain model universal circuit evaluator* which accepts a pair of inputs (C, x) where C is a *plain model* circuit and x is a string in the domain of C, and $\mathsf{Univ}(C, x)$ outputs $C(x)$.

Many primitives of interest in this work can be seen as the universal member of an indexed family of primitives. For example, in the case of witness encryption, its definition can be written using a generic witness relation V that checks the relation between the instance and the witness. However, in the *standard* definition of the witness encryption, the relation V is universal; namely, the relation between witness w and attribute a is verified by running a as a circuit over w and outputting the first bit of this computation. In order to generalize the notion of universal members of indexed primitives (i.e., special primitives for short) we need the following definition.

The following definition defines primitives that have a "recursive" nature using an *oracle* algorithm $V^{(\cdot)}$ that calls the implementation F of the primitive itself.

Definition 4.3 ($V^{(\cdot)}$-Indexed Primitives)**.** For a $\mathcal{W}$-indexed family $\mathcal{P}[\mathcal{W}] = \{(\mathcal{F}[W], \mathcal{R}[W])\}_{W\in\mathcal{W}}$, and an oracle algorithm $V^{(\cdot)}$, consider the following primitive $\mathcal{Q} = (\mathcal{F}_\mathcal{Q}, \mathcal{R}_\mathcal{Q})$ defined as follows. For all $W \in \mathcal{W}$ and $F \in \mathcal{F}[W]$, if $W = V^F$, then we include F in $\mathcal{F}_\mathcal{Q}$, and for any such W, F, if $(F, A) \in \mathcal{R}[W]$, then we include (F, A) in $\mathcal{R}_\mathcal{Q}$ as well. Even though $V^{(\cdot)}$ is not a fixed algorithm (and recursively depends on the implementation function), and although $\mathcal{Q}$ might not be a member of the indexed family $\mathcal{P}[\mathcal{W}]$ for any $W \in \mathcal{W}$, we still call $\mathcal{Q}$ the $V^{(\cdot)}$*-indexed* primitive defined by (or, from) $\mathcal{P}[\mathcal{W}]$ and denote it as $\mathcal{P}[V^{(\cdot)}]$.

Definition 4.4 (Monolithic Extension of Indexed Primitives)**.** For any indexed family $\mathcal{P}[\mathcal{W}]$ and $V \in \mathcal{W}$, we call the $V^{(\cdot)}$-indexed primitive $\mathcal{P}[V^{(\cdot)}]$ the *monolithic extension* (or simply the *extension*) of the V-indexed primitive $\mathcal{P}[V]$ (both with respect to the family $\mathcal{P}[\mathcal{W}]$). Whenever $\mathcal{P}[\mathcal{W}]$ is clear form the context, we simply call $\mathcal{P}[V^{(\cdot)}]$ the monolithic extension of $\mathcal{P}[V]$.

We are particularly interested in monolithic extensions of the Univ-indexed primitive $\mathcal{P}[\mathsf{Univ}]$ from a family of primitives $\mathcal{P}[\mathcal{W}]$, which is automatically defined as $\mathcal{P}[\mathsf{Univ}^{(\cdot)}]$. However, note that in order to even define $\mathcal{P}[\mathsf{Univ}^{(\cdot)}]$ we always need to first define the corresponding indexed family of primitives $\mathcal{P}[\mathcal{W}]$.

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

Remark 4.5 (Non-black-box relation between the universal and extended primitives)**.** Even though the universal selection $\mathcal{P}[\mathsf{Univ}]$ and its monolithic extension $\mathcal{P}[\mathsf{Univ}^{(\cdot)}]$ (with respect to family $\mathcal{P}[\mathcal{W}]$) are tightly related, they are indeed different cryptographic primitives according to Definition 3.12, e.g., because their input formats could be potentially different. However, for 'natural' primitives, and in particular for all the primitives studied in this work, one can get one from the other one in a *non-black-box* way. The intuition is that one can us the code of $\mathcal{P}[\mathsf{Univ}]$ to open up the gates planted in the circuits given to the oracle-aided universal circuit $\mathsf{Univ}^{(\cdot)}$ used in the definition of $\mathcal{P}[\mathsf{Univ}^{(\cdot)}]$ (see example below).

Special 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 $V(a, w)$ 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 $V \in \mathcal{W}$ which is 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 Univ-indexed primitive WE[Univ] with respect to the family WE[$\mathcal{W}$], and the following will be exactly the definition of the monolithic extension of WE[Univ], which we simply call extended WE.

Definition 4.6 (Extended Witness Encryption)**.** Let $\mathsf{V}^{(\mathrm{Enc},\mathrm{Dec})}(w, a) := \mathsf{Univ}^{(\mathrm{Enc},\mathrm{Dec})}(a, w)$ be the 'universal circuit-evaluator' Turing machine as defined in Definition 4.2. The *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 $m \in \{0,1\}^*$ or $\perp$.
- Correctness and security are defined similarly to Definition 3.3. But the key point is that here the relation $\mathsf{V}^{(\mathrm{Enc},\mathrm{Dec})}$ is somehow recursively depending on the $(\mathrm{Enc}, \mathrm{Dec} = \mathrm{Dec}_{\mathsf{V}})$ on smaller input lengths (and so it is well defined).

Definition 4.7 (Extended Variants of More Primitives)**.** Extended variants for the following primitives are defined similarly to Definition 4.6 by allowing their special subroutine to be based on an oracle algorithm that is the universal circuit evaluator Univ:

- For attribute-based or predicate encryption, we allow the predicate algorithm $\mathsf{P}(k, a)$ to be an oracle algorithm that interprets k as an oracle-aided circuit and runs a on k.
- For FHE and Spooky encryption we allow the evaluation function $\mathsf{F}(f, m_1, ..., m_t)$ to be an oracle algorithm that that interprets f as an oracle-aided circuit and runs f on $(m_1, ..., m_t)$.

4.2 Defining Monolithic Constructions Using Monolithic Extensions

We are finally ready to define our monolithic framework. Here we assume that for a primitive $\mathcal{P} = \mathcal{P}[V] \in \mathcal{P}[\mathcal{W}]$ we have already defined its monolithic extension $\widetilde{\mathcal{P}} = \mathcal{P}[V^{(\cdot)}]$ (with respect to $\mathcal{P}[\mathcal{W}]$).

Definition 4.8 (Monolithic Constructions)**.** Suppose $\mathcal{Q}$ is a primitive and $\widetilde{\mathcal{P}}$ is the monolithic extension of the primitive $\mathcal{P}$. Any fully black-box construction for $\mathcal{Q}$ from $\widetilde{\mathcal{P}}$ (i.e. the monolithic extension of $\mathcal{P}$) is called a *monolithic* construction of $\mathcal{Q}$ from $\mathcal{P}$.

Examples. Below are some examples of *non-black-box* constructions in cryptography that fall into the monolithic framework of Definition 4.8.

- Gentry's bootstrapping construction [Gen09a] plants FHE's own decryption in a circuit for the evaluation subroutine. This trick falls into the monolithic framework since planting gates inside evaluation circuits is allowed.
- The construction of IO from functional encryption by [AJ15, BV15] uses the encryption oracle of the functional encryption scheme inside the functions for which decryption keys are issued. Again, such *non-black-box* technique does fall into our monolithic framework.

Definition 4.9 (Formal Definition of Monolithic Constructions for Specific Primitives)**.** Let $\mathcal{P}$ be any of the following primitives: attribute, predicate, functional, or witness encryption, FHE, multi-key FHE, or spooky encryption. Then a monolithic construction using $\mathcal{P}$ is defined by first defining the monolithic extension $\widetilde{\mathcal{P}}$ for primitive $\mathcal{P}$ according to Definitions 4.6 and 4.7, then applying Definition 4.8 (which is based on previously defined notion of *fully* black-box constructions).

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

Lemma 4.10 (Composing Monolithic Constructions)**.** *Suppose $\mathcal{P}$, $\mathcal{Q}, \mathcal{R}$ are cryptographic primitives. If there is a monolithic construction of the monolithic extension $\widetilde{\mathcal{Q}}$ from $\mathcal{P}$ and if there is a monolithic construction of $\mathcal{R}$ from $\mathcal{Q}$, then there is a monolithic construction of $\mathcal{R}$ from $\mathcal{P}$.*

Proof. Since there is a monolithic construction of $\mathcal{R}$ from $\mathcal{Q}$, by Definition 4.8 it means that there exists a monolithic 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 4.8, for any monolithic extension of $\mathcal{Q}$, and in particular $\widetilde{\mathcal{Q}}$, there is a fully black-box construction of $\widetilde{\mathcal{Q}}$ from some monolithic 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 4.8) means we have a monolithic construction of $\mathcal{R}$ from $\mathcal{P}$. □

Getting more separations. A corollary of Lemma 4.10 is that if one proves: (a) There is no monolithic construction of $\mathcal{R}$ from $\mathcal{P}$ and (b) there *is* a monolithic construction of any monolithic extension $\widetilde{\mathcal{R}}$ (of $\mathcal{R}$) from $\mathcal{Q}$, then these two together imply that: there is no monolithic 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.

5 Separating IO from Instance Revealing Witness Encryption

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

Theorem 5.1. *Assume the existence of one-way functions and that* $\mathbf{NP} \not\subseteq \mathbf{coAM}$*. Then there exists no monolithic construction of indistinguishability obfuscation from witness encryption .*

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 4.10, this would also imply a separation of IO from extended ABE.

Definition 5.2 (V-Indexed Extractable Instance-Revealing Witness Encryption)**.** Let $\mathcal{W}$ be the set of all predicates. For any given security parameter κ and any $\mathsf{V} \in \mathcal{W}$, a V-*indexed extractable instance-revealing witness encryption* scheme EIRWE[V] consists of three PPT algorithms $(\mathrm{Enc}, \mathrm{Rev}, \mathrm{Dec})$ 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 $m' \in \{0,1\}^*$.

A V-indexed extractable instance-revealing witness encryption family satisfies the following completeness and security properties:

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

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

- **Instance-revealing correctness.** For any security parameter κ and any (a, m) we have:

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

  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 $a \in \{0,1\}^*$, and any $m_0 \neq m_1$ of the same length $|m_0| = |m_1|$, if:

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

  then

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

Definition 5.3 (Extended Extractable Instance-Revealing Witness Encryption)**.** An *extended extractable instance-revealing witness encryption* scheme (ex-EIRWE) is defined as the monolithic extension EIRWE[$\mathsf{V}^{(\cdot)}$] (see Definition 4.4) of the V-indexed EIRWE scheme EIRWE[V] where V is the oracle-aided universal circuit evaluator $\mathsf{Univ}^{(\cdot)}$.

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

Since extended EIRWE implies witness encryption as defined in Definition 3.3, Theorem 5.1 trivially follows from Theorem 5.4, and thus for the remainder of this section we will focus on proving Theorem 5.4.

5.1 Overview of Proof Techniques

To prove Theorem 5.4, we will apply Lemma 3.26 for the idealized extended IRWE model Θ (formally defined in Section 5.2) to prove that there is no black-box construction of IO from any primitive $\mathcal{P}$ that can be oracle-fixed constructed (see Definition 3.16) 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 5.3 and the second task is proven in Section 5.4. By Lemma 3.26, this would imply the separation result of IO from $\mathcal{P}$ and prove Theorem 5.4.

Our oracle, which is more formally defined in Section 5.2, 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 3.23 in this idealized model.

5.2 The Ideal Model

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

Definition 5.5 (Random Instance-Revealing Witness Encryption Oracle)**.** Let V be an oracle-aided universal circuit-evaluator Turing machine (as defined in Definition 4.2) that takes as input (a, w) where a is a Boolean oracle circuit. 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 $\Theta = \{\Theta_n\}_{n\in\mathbb{N}}$.

- $\mathrm{Enc}\colon \{0,1\}^n \to \{0,1\}^{2n}$ is a random injective function.
- $\mathrm{Rev}\colon \{0,1\}^{2n} \to \{0,1\}^* \cup \bot$ 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.
- $\mathrm{Dec}_{\mathsf{V}}\colon \{0,1\}^s \to \{0,1\}^n \cup \{\bot\}$: 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 (a, w). More formally, do as follow:
  1. If $\nexists\, x$ such that $\mathrm{Enc}(x) = c$, output $\bot$. Otherwise, continue to the next step.
  2. Find x such that $\mathrm{Enc}(x) = c$.

3. If $\mathsf{V}^{\Theta}(a, w) = 0$ output $\perp$. Otherwise, output $x = (a, m)$.

We define a query-answer pair resulting from query q to subroutine $T \in \{\text{Enc}, \text{Dec}, \text{Rev}\}$ 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 $\Theta = (\text{Enc}, \text{Rev}, \text{Dec}_{\mathsf{V}})$ to refer to Θ_n for a fixed n.

Remark 5.6. 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*.

5.3 Witness Encryption exists relative to Θ

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

Lemma 5.7. *There exists a correct and subexponentially-secure oracle-fixed implementation (Definition 3.16) of extended extractable instance-revealing WE 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 extended EIRWE. We give the definition of the indexed and extended variant of this primitive followed by a construction.

Definition 5.8 (V-Indexed Extractable One-Way Witness Encryption)**.** Let $\mathcal{W}$ be the set of all predicates. For any given security parameter κ and any $V \in \mathcal{W}$, a V*-indexed extractable one-way witness encryption* scheme EOWE[V] consists of the following PPT algorithms $(\text{Enc}, \text{Rev}, \text{Dec}_{\mathsf{V}})$ defined as follows:

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

A V-indexed extractable one-way witness encryption scheme satisfies the same correctness properties as Definition 5.3 but the extractability property is replaced with the following:

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

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

  then

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

Definition 5.9 (Extended Extractable One-Way Witness Encryption)**.** An *extended extractable one-way witness encryption* scheme (ex-EOWE) is defined as the monolithic extension EOWE[$\mathsf{V}^{(\cdot)}$] (see Definition 4.4) of the V-indexed EOWE scheme EOWE[V] where V is the oracle-aided universal circuit evaluator $\mathsf{Univ}^{(\cdot)}$.

Construction 5.10 (Extended Extractable One-Way Witness Encryption)**.** For any security parameter κ and oracle Θ sampled according to Definition 5.5, we will implement an extended EOWE scheme using $\Theta = (\mathrm{Enc}, \mathrm{Rev}, \mathrm{Dec}_{\mathsf{V}})$ (where recall that V in Θ is defined as to be the universal circuit evaluator) as follows:

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

Remark 5.11 (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 thoerem [GL89]. Namely, to encrypt a one-bit message b under some attribute a, we would output the ciphertext $c = (\mathrm{Enc}(a, r_1), r_2, \langle r_1, r_2\rangle \oplus b)$ 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 $r_1 = \mathrm{Dec}(w, y_1)$, 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 one-way function $\mathrm{Enc}'_a(r_1, r_2) := (\mathrm{Enc}(a, r_1), r_2)$, we have $(\mathrm{Enc}(a, r_1), r_2, \langle r_1, r_2\rangle \oplus 0) \approx (\mathrm{Enc}(a, r_1), r_2, \langle r_1, r_2\rangle \oplus 1)$ for any fixed a.

Lemma 5.12. *Construction 5.10 is a correct and subexponentially-secure oracle-fixed implementation (Definition 3.16) of extended extractable one-way WE 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 Construction 5.10 (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 $\Theta = (\mathrm{Enc}, \mathrm{Rev}, \mathrm{Dec}_{\mathsf{V}})$ 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 $c^* \leftarrow \mathrm{WEnc}(a, m, 1^\kappa)$ is the challenge. Queries issued by A are handled by E as follows:

- To answer any query $\mathrm{Enc}(x)$ 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 $a \neq \bot$, 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 $a = \bot$, it returns $\bot$ 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 5.13. *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 5.14. *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 5.13, 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 5.10 is a correct implementation. Furthermore, by Lemma 5.13, it satisfies the extractability property. Thus, this concludes the proof of Lemma 5.12. □

Proof of Lemma 5.7. The existence of extractable instance-revealing witness encryption in the Θ oracle model follows from Lemma 5.12 and Remark 5.11. □

5.4 Compiling Out Θ from IO

In this section, we show a simulatable compiler for compiling out Θ. We adapt the approach outlined in Section 5.1 to the ideal IRWE oracle $\Theta = (\mathrm{Enc}, \mathrm{Rev}, \mathrm{Dec}_{\mathsf{V}})$ while making use of Lemma 3.25, 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 [CKP15] to compile out the random oracle R and get an obfuscator O' in the plain-model. Since we are applying this lemma only a constant number of times (in fact, just twice), security should still be preserved. Specifically, we will prove the following claim:

Lemma 5.15. *Let $R \sqsubseteq \Theta$ be a random oracle where "$\sqsubseteq$" denotes a sub-model relationship (see Definition 3.22). 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.*
- *[CKP15] For any IO in the random oracle R model, there exists a simulatable compiler with correctness error $\epsilon < 1/200$ for it that outputs a new obfuscator in the plain model.*

Proof. The second part of Lemma 5.15 follows directly by [CKP15], and thus we focus on proving the first part of the claim. Before we start describing the compilation process, we present the following definition of canonical executions that is a property of algorithms in this ideal model and dependent on the oracle being removed.

Definition 5.16 (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 polynomial query algorithm).

Definition 5.17 (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 5.16, 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 5.16.

5.4.1 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.

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

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 the answer $\rho_q \leftarrow \texttt{EmulateCall}^{R}(Q_O \cup Q_B, q)$ 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 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 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 not be determined by Q_B in the real experiment and will be answered $\perp$.

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 5.18. $\Pr_x[E(x)] \leq \epsilon$.

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 attribute a) via Rev queries during the learning.

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. □

6 Separating IO from Instance Hiding Witness Encryption

In this section, we formally prove our second main separation theorem which states that there is no fully black-box construction of indistinguishability obfuscation from any *extended* predicate encryption scheme.

Theorem 6.1. *Assume that* $\mathbf{NP} \not\subseteq \mathbf{coAM}$ *and that one-way functions exist. Then there exists no monolithic construction of indistinguishability obfuscation (IO) from predicate encryption (PE).*

In fact, we prove a stronger result by showing a separation of IO from a more generalized primitive, which we call extended *extractable instance-hiding* witness encryption, that implies extended predicate encryption and is a stronger variant of witness encryption.

Definition 6.2 (Extended Extractable Instance-Hiding Witness Encryption (ex-EIHWE))**.** Let V be a universal circuit-evaluator as defined in Definition 4.6. For any given security parameter κ, an *extended extractable instance-hiding witness encryption* scheme consists of two PPT algorithms $P = (\mathrm{Enc}, \mathrm{Dec}_{\mathsf{V}})$ 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 $c \in \{0,1\}^*$.
- $\mathrm{Dec}_{\mathsf{V}}(w, c)$: given ciphertext c and "witness" string w, it outputs a message $m' \in \{0,1\}^*$.

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

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

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

- **Instance-hiding extractability.** For any PPT adversary A and polynomial $p_1(.)$, there exists a PPT (black-box) straight-line extractor E and a polynomial function $p_2(.)$ such that the following holds. For security parameter κ, for all a_0, a_1 of the same length $|a_0| = |a_1|$ and any $m_0 \neq m_1$ of the same length $|m_0| = |m_1|$, if:

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

then:

$$\Pr[E^A(a_0, a_1) = w \wedge \exists\, b \in \{0,1\}\ s.t.\ \mathsf{V}^P(w, a_b) = 1] \geq \frac{1}{p_2(\kappa)}.$$

Generalizing the verification algorithm. Note that while the standard witness encryption scheme defines the verification algorithm as operating on an witness-instance pair (w, a), since we are elevating the security to be instance-hiding, we can generalize the verifier to allow it to accept and act on not only the instance a but also the message m as input. That is, the verifier V in our definition of instance-hiding witness encryption would accept (w, x) instead of just (w, a) where $x = (a, m)$. As a result, whenever it is unambiguous, we would often refer to x as the instance.

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

Theorem 6.3. *Assume that* $\mathbf{NP} \not\subseteq \mathbf{coAM}$ *and that one-way functions exist. Then there exists no* monolithic *construction of indistinguishability obfuscation from instance-hiding witness encryption.*

In Section 8.1, we prove that extended EIHWE implies extended predicate encryption. As a result, Theorem 6.1 would follow from Theorem 6.3, Lemma 8.1 and Lemma 4.10 (the transitivity lemma). Hence, for the remainder of this section we will focus on proving Theorem 6.3.

6.1 Overview of Proof Techniques

To prove Theorem 6.3, we will apply Lemma 3.26 for the idealized IHWE model Θ (formally defined in Section 6.2) to prove that there is no black-box construction of IO from any primitive $\mathcal{P}$ that can be oracle-fixed constructed from Θ. In particular, we will do so for $\mathcal{P}$ that is the extended IHWE 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 6.3 and the second task is proven in Section 6.4. By Lemma 3.26, this would imply the separation result of IO from $\mathcal{P}$ and prove Theorem 6.3.

Our oracle, which is more formally defined in Section 6.2, resembles an idealized version of a (instance-hiding) witness encryption scheme, which makes the construction of extended IHWE straightforward. As a result, the main challenge lies in showing a simulatable compilation procedure for IO that satisfies Definition 3.23 in this idealized model, and therefore, it is instructive to look at how the compilation process works and what challenges are faced with dealing with oracle Θ.

6.1.1 High-level Compiler Structure and Challenges

Recall that the solution adopted for the (extended) IRWE oracle was to canonicalize the obfuscated circuits such that queries made by the verification algorithm through decryption queries are made manifest so that they can be publicized without affecting security. However, we shall see that this is not enough to argue that (approximate) correctness is maintained.

Challenge faced with (instance-hiding) witness encryption. Here we are again faced with a problem when we try to emulate decryption queries. Very much like the setting of extended IRWE, since Θ represents an idealized extended primitive, the verification algorithm V associated with the language defined by Θ might make oracle calls during the learning process. However, precisely due to the instance-hiding property, one cannot canonicalize the obfuscated circuits as before since we cannot efficiently extract the attribute or the message to run $\mathsf{V}^{\Theta}(x, w)$ and discover the queries asked. This creates the possibility that the execution $B'(x)$ might call such a hidden query and emulating it incorrectly.

Resolving the challenge. While a full canonicalization is not possible here, we still perform it partially. Specifically, whenever the obfuscated circuit calls a decryption query $\mathrm{Dec}(w, c)$ and the underlying message x is successfully decrypted, the obfuscated circuit can now run $\mathsf{V}^{\Theta}(x, w) = 1$ to reveal the queries asked. However, it is possible that the decryption fails, in which case we may have some hidden queries Q_{V} asked by an underlying $\mathsf{V}^{\Theta}(x, w) = 0$ that cannot be revealed or learned. Nonetheless, we argue that the gathered information is enough for an approximately correct emulation of the plain-model obfuscated code.

6.2 The Ideal Model

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

Definition 6.4 (Random Instance-Hiding Witness Encryption Oracle)**.** Let V be a PPT universal circuit-evaluator as defined in Definition **??** that takes as input (w, x) where $x \in \{0,1\}^n$ and outputs $b \in \{0,1\}$. We define the following *random instance-hiding witness encryption* (rIHWE) oracle $\Theta = (\mathrm{Enc}, \mathrm{Dec}_{\mathsf{V}})$ 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. We will use the Enc oracle to encrypt a message $x \in \{0,1\}^n$ to obtain a ciphertext $c \in \{0,1\}^{2n}$.
- $\mathrm{Dec}_{\mathsf{V}}\colon \{0,1\}^s \mapsto \{0,1\}^n \cup \{\perp\}$: Given $w \in \{0,1\}^k, c \in \{0,1\}^{2n}$ as inputs and $s = k + 2n$, $\mathrm{Dec}(w, c)$ allows us to decrypt the ciphertext c and get x as long as the predicate test is satisfied on (w, x). More formally, do as follow:
  1. If $\nexists\, x$ such that $\mathrm{Enc}(x) = c$, output $\perp$. Otherwise, continue to the next step.
  2. Find x such that $\mathrm{Enc}(x) = c$.
  3. If $\mathsf{V}^{\Theta}(w, x) = 0$ output $\perp$. Otherwise, output x.

We define a query-answer pair resulting from query q to subroutine $T \in \{\mathrm{Enc}, \mathrm{Dec}\}$ 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, $\Theta = (\mathrm{Enc}, \mathrm{Dec}_{\mathsf{V}})$ refers to Θ_n for a fixed n.

6.3 Instance-Hiding Witness Encryption Exists Relative to Θ

In this section, we show how to construct a semantically-secure extended EIHWE with corresponding verification algorithm V relative to $\Theta = (\mathrm{Enc}, \mathrm{Dec}_{\mathsf{V}})$.

Construction 6.5 (Extended Extractable Instance-Hiding Witness Encryption)**.** Let V be a universal circuit-evaluator as defined in Definition **??**. For any security parameter κ and oracle Θ_κ sampled according to Definition 6.4, we will implement an extended EIHWE scheme P as follows:

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

Lemma 6.6. *Construction 6.5 is a correct and subexponentially-secure oracle-fixed implementation (Definition 3.16) of extended extractable instance-hiding WE in the ideal Θ oracle model.*

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 distinguish between encryptions of two different messages with non-negligible advantage then there exists a (fixed) deterministic straight-line extractor E with access to $\Theta = (\text{Enc}, \text{Dec}_{\mathsf{V}})$ that can find the witness for the instance of the challenge ciphertext.

Suppose A is an adversary in the indistinguishability game with success probability $1/2 + \epsilon$. Then the extractor E would works as follows: given (a_0, a_1) as input and acting as the challenger for adversary A, it runs $A^{\Theta}(1^{\kappa}, c^*)$ where $c^* \leftarrow \text{WEnc}(a_b, m_b, 1^{\kappa})$ is the challenge ciphertext. 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 $\text{Dec}(w, c)$ for $c \neq c^*$ asked by A, the extractor will instead attempt to simulate this query for A instead of asking Θ directly. In particular, it first verifies whether there exists a query-answer pair $(x \mapsto c)_{\text{Enc}}$ that was generated by A previously. If such a query-answer pair exists then the extractor will run $\mathsf{V}^{\Theta}(w, x) = \beta$ and if $\beta = 1$ it returns to A the answer x. If such a query-answer pair does not exist or $\beta = 0$ then it returns $\perp$ to A. Note that, while running $\mathsf{V}^{\Theta}(w, x)$, the extractor will also need to answer the queries asked by V similar to how it did for A. While handling the queries made by A, if a decryption query $\text{Dec}_{\mathsf{V}}(w, c^*)$ for the challenge ciphertext is issued by A, the extractor will pass this query to 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 6.7. *For any PPT adversary A, pair of instances a_0, a_1 of the same length $|a_0| = |a_1|$ and any $m_0 \neq m_1$ of the same length $|m_0| = |m_1|$, if there exists a non-negligible function $\epsilon(.)$ such that:*

$$\Pr\left[A^{\Theta}(1^{\kappa}, c^*) = b \mid b \xleftarrow{\$} \{0,1\}, c^* \leftarrow \text{WEnc}(a_b, m_b, 1^{\kappa})\right] \geq \frac{1}{2} + \epsilon(\kappa) \tag{3}$$

then there exists a PPT extractor E such that:

$$\Pr\left[E^{\Theta, A}(a_0, a_1) = w \wedge \ \mathsf{V}^{\Theta}(w, a_b) = 1 \mid b \xleftarrow{\$} \{0,1\}\right] \geq \epsilon(\kappa) - \text{negl}(\kappa). \tag{4}$$

Let A be an adversary satisfying Equation (3) above and let AdvWin be the event that A succeeds in the distinguishing game. Furthermore, let ExtWin be the event that the extractor succeeds in extracting a witness (as in Equation (4) above). Observe that:

$$\Pr_{\Theta,b,r}[\overline{\mathsf{ExtWin}}] \leq \Pr_{\Theta,b,r}[\overline{\mathsf{ExtWin}} \wedge \mathsf{AdvWin}] + \Pr_{\Theta,b,r}[\overline{\mathsf{AdvWin}}]$$

Since $\Pr[\mathsf{AdvWin}] \geq 1/2 + \epsilon$ for some non-negligible function ϵ, it suffices to show that $\Pr[\overline{\mathsf{ExtWin}} \wedge \mathsf{AdvWin}]$ is negligibly close to $1/2$. Note that, by our construction of extractor E, this event is equivalent to saying that the adversary succeeds in the distinguishing game but never asks a query of the form $\text{Dec}_{\mathsf{V}}(w, c^*)$ 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 only a negligible factor over the trivial advantage. That is, we will prove the following claim:

Claim 6.8. *For any negligible* ϵ, $\Pr_{\Theta}\left[\Pr_{b,r}[\mathsf{Win}] \geq 1/2 + \epsilon\right] \leq \mathrm{negl}(\kappa)$.

Proof. We first show that we can construct an adversary $\widehat{A}$ that has the same winning advantage as A but does not ask decryption queries (only encryption queries). 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. Given the challenge c^*, the new adversary $\widehat{A}$ executes $A^{\Theta}(c^*)$ and answers the queries as follows: if the query is an encryption query then it is directly forwarded to Θ to get the answer. If it is a $\mathrm{Dec}(w, c')$ query for some $c' \neq c^*$ then it will check if $\mathrm{Enc}(x') = c'$ was asked by A before in which case it runs $\mathsf{V}^{\Theta}(w, x') = \beta$ and returns the answer x' if and only if $\beta = 1$. If no such encryption was asked then event Bad has happened and we abort the execution. If a query $\mathrm{Dec}(w, c^*)$ was asked then we return $\perp$ immediately since $\overline{\mathsf{ExtWin}}$ does not happen and therefore the Dec is expected to fail anyway. Thus, as long as Bad does not happen, the advantage of $\widehat{A}$ in winning the distinguishing game is at least the advantage of A.

It is straightforward to show, by a standard application of an averaging argument, that the expression $\Pr_{\Theta}\left[\Pr_{b,r}[\mathsf{Bad}] \geq \epsilon/3\right] \leq \mathrm{negl}(\kappa)$. Specifically, we know that 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, $\Pr_{\Theta,b,r}[\mathsf{Bad}] \leq 1/2^n$, so by an averaging argument $\Pr_{\Theta}\left[\Pr_{b,r}[\mathsf{Bad}] \geq \epsilon/3\right] \leq \Pr_{\Theta}\left[\Pr_{b,r}[\mathsf{Bad}] \geq 2^{-n/2}\right] \leq 2^{-n/2}$.

We now proceed to show that $\widehat{A}$ only succeeds with negligible advantage while asking only encryption queries. Define Hit to be the event that $\widehat{A}$ happens to ask $\mathrm{Enc}(x^*) = c^*$. Then:

$$\begin{aligned}
\Pr_{\Theta}\left[\Pr_{b,r}[\mathsf{Win}] \geq \frac{1}{2} + \epsilon\right] &\leq \Pr_{\Theta}\left[\Pr_{b,r}[\mathsf{Win} \wedge \overline{\mathsf{Bad}} \wedge \overline{\mathsf{Hit}}] + \Pr_{b,r}[\mathsf{Bad}] + \Pr_{b,r}[\mathsf{Hit}] \geq \frac{1}{2} + \epsilon\right] \\
&\leq \Pr_{\Theta}\left[\Pr_{b,r}[\mathsf{Win} \wedge \overline{\mathsf{Bad}} \wedge \overline{\mathsf{Hit}}] \geq \frac{1}{2} + \frac{\epsilon}{3} \vee \Pr_{b,r}[\mathsf{Bad}] \geq \frac{\epsilon}{3} \vee \Pr_{b,r}[\mathsf{Hit}] \geq \frac{\epsilon}{3}\right] \\
&\leq \Pr_{\Theta}\left[\Pr_{b,r}[\mathsf{Win} \wedge \overline{\mathsf{Bad}} \wedge \overline{\mathsf{Hit}}] \geq \frac{1}{2} + \frac{\epsilon}{3}\right] + \Pr_{\Theta}\left[\Pr_{b,r}[\mathsf{Bad}] \geq \frac{\epsilon}{3}\right] \\
&\qquad + \Pr_{\Theta}\left[\Pr_{b,r}[\mathsf{Hit}] \geq \frac{\epsilon}{3}\right].
\end{aligned}$$

We can bound the event Hit from happening since it is the event that we invert the image of a random injective function. This can be done over the randomness of the oracle then, using an averaging argument, deduce that the probability that Hit happens for a non-negligible fraction of oracles Θ is negligible.

We will thus focus on proving the first term. In doing so, we will reduce the problem of indistinguishability to that of predicting the output of a random Boolean function on a random point in the domain of this function. We then prove that the latter problem is hard using a compression technique which allows to argue that an adversary that can win the prediction game with non-negligible advantage can only do so for a negligible fraction of the oracles. We will make use of the following lemma that shows the existence of a randomized compression technique for random Boolean functions using adversaries against the prediction game.

Lemma 6.9 (Lemma 10.4 [DTT10])**.** *Let A be an oracle algorithm that makes q queries to some fixed oracle predicate $p : \{0,1\}^n \to \{0,1\}$, never queries the oracle on its input and satisfies the following for some ϵ:*

$$\Pr[A^p(x) = p(x)] \geq 1/2 + \epsilon$$

then there exists a randomized encoder E and a randomized decoder D such that:

$$\Pr_r[D(E(p,r),r) = p] \geq \epsilon/q \quad \textit{and} \quad |E(p,r)| \leq 2^n - \epsilon^2 2^n/q.$$

For any fixed pair (a_0, a_1) and given adversary $\widehat{A}$ that wins in the indistinguishability game without asking any decryption queries, we can construct a new oracle-aided adversary $\widetilde{A}$ that aims to win in the experiment $\text{Exp}_{\widetilde{A}}^{P}(1^\kappa, a_0, a_1)$ as defined in Figure 4 without querying the oracle on its input and without asking any decryption queries. The adversary $\widetilde{A}$ will have access to oracle Θ and P^Θ, which is defined as follows:

$$P(a_0, a_1, r, c_0, c_1) = \begin{cases} 0 \text{ if } c_0 = \text{Enc}(a_0||0||r) \text{ and } c_1 = \text{Enc}(a_1||1||r) \\ 1 \text{ if } c_0 = \text{Enc}(a_1||1||r) \text{ and } c_1 = \text{Enc}(a_0||0||r) \\ \perp \text{ otherwise} \end{cases}$$

Experiment $\text{Exp}_A^P(1^\kappa, a_0, a_1)$:

1. $r \xleftarrow{\$} \{0,1\}^{n/2-1}, b \xleftarrow{\$} \{0,1\}$
2. $c_0^* \leftarrow \text{Enc}(a_b||b||r), c_1^* \leftarrow \text{Enc}(a_{1-b}||1-b||r)$
3. $b' \leftarrow A^{P,\Theta}(r, c_0^*, c_1^*)$ where A cannot query P on $(a_0, a_1, r, c_0^*, c_1^*)$ or $(a_0, a_1, r, c_1^*, c_0^*)$
4. Output 1 if $b = b'$ and 0 otherwise.

Figure 4: The Exp_A^P Experiment

The adversary $\widetilde{A}$, given (r, c_0^*, c_1^*), would execute $b' \leftarrow \widehat{A}(c_0^*)$ and outputs b' as its answer. We can modify $\widetilde{A}$ so that whenever it issues a query to $\text{Enc}(a||b||r) = c_b$, it would also call $\text{Enc}(a||1-b||r) = c_{1-b}$ followed by a call to $P(a, a, r, c_0, c_1)$. This ensures that any encryption query to Θ is translated into a query to P.

Note that P can be interpreted as an alternative description for Θ. That is, given $P : \{0,1\}^m \to \{0,1\}$ where $m = 11n/2 - 1$, one can reconstruct Θ on any point in its domain. Define $P_{a_0,a_1} := P(a_0, a_1, ., ., .)$ to be the function P restricted to the attributes a_0 and a_1. By Lemma 6.9, we can encode P_{a_0,a_1} using at most $m - \epsilon^2 m/9q$ bits where q is the number of queries that A makes. Thus, we have compressed the entire oracle by saving $\alpha = \epsilon^2 m/9q$ bits. Hence, assuming that $\epsilon = \text{negl}(\kappa)$ and $q = \text{poly}(\kappa)$ we find that the fraction of oracles for which $\widetilde{A}$ can win on is at most $1/2^\alpha = \text{negl}(\kappa)$.

□

Proof of Lemma 6.6. It is clear that the Construction 6.5 is correct. Furthermore, by Lemma 6.7, it also satisfies the extractability property. □

6.4 Compiling Out Θ from IO

In this section, we show a simulatable compiler for compiling out Θ. We formalize the approach outlined in Section 6.1 while making use of Lemma 3.25, 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 [CKP15] to compile out the random oracle R and get an obfuscator O' in the plain-model. Since we are applying this lemma only a constant number of times (in fact, just twice), security should still be preserved. Specifically, we will prove the following claim:

Lemma 6.10. *Let $R \sqsubseteq \Theta$ be a random oracle where "$\sqsubseteq$" denotes a sub-model relationship (see Definition 3.22). Then the following holds:*

- *For any IO in the Θ ideal model, there exists a simulatable compiler for it with correctness error $\epsilon < 1/200$ that outputs a new obfuscator in the random oracle R model.*
- *[CKP15] For any IO in the random oracle R model, there exists a simulatable compiler for it with error correctness $\epsilon < 1/200$ that outputs a new obfuscator in the plain model.*

Proof. The second part of Lemma 6.10 follows directly from the previous result of [CKP15], and thus we focus on proving the first part of the claim. Before we start describing the compilation process, we present the following definition of canonical executions that is a property of algorithms in this ideal model and dependent on the oracle being removed, then we describe the notation and terminology used throughout the proof.

Definition 6.11 (Canonical Executions)**.** We define an oracle algorithm A^{Θ} relative to rIHWE to be in canonical form if right after asking any $\mathrm{Dec}_{\mathsf{V}}(w, c) = x$ where $x \neq \perp$, A would also additionally ask the query $\mathrm{Enc}(x)$ followed by directly executing $\mathsf{V}^{\Theta}(w, x)$ on its own. Note that any algorithm A can be easily modified into a canonical form by increasing its query complexity at most by a polynomial factor (since V is a PPT algorithm).

Definition 6.12 (Query Types)**.** For any canonical oracle algorithm A with access to a rIHWE 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, while all direct queries are visible to A, an indirect query q may or not may visible to A; an indirect query q is visible to A if it runs V^{Θ} and q is asked by V then forwarded to A (where q was not previously asked by A in an explicit manner).

Transcripts. We use bold fonts (e.g., $\mathbf{T}$) to denote the distribution from which we sample an actual transcript $T \leftarrow \mathbf{T}$. For any transcript T of some oracle algorithm, we define $U(T)$ to be the set of query-answer pairs asked of Θ during the generation of T and define $Q(T)$ to be the projection of $U(T)$ onto the first element, which represents the set of queries only.

We now proceed to present the construction of the random-oracle model obfuscator that, given an indistinguishability obfuscator $O = (iO, Ev)$ in the Θ model, would compile out and emulate

queries to Dec while forwarding any Enc queries to R. Throughout this process, we assume, without loss of generality, that the ideal-model obfuscator iO and the ideal-model evaluation of the obfuscation Ev are both in canonical form according to Definition 6.11. For these canonical algorithms, let $\ell_O, \ell_B = \text{poly}(n)$, respectively, be the number of queries asked by the ideal-model obfuscator iO and the evaluation Ev to the rIHWE oracle Θ.

Algorithm 2: `EmulateCall`

Input: Query-answer set U and query q
Oracle: Random oracle R
Output: (ρ_q, W_H) where ρ_q is a query-answer pair containing the answer of query q and W_H is the set of indirect query-answer pairs (those asked by V)

Begin.
Initialize $W_H = \varnothing$
if q is a query of type $\text{Enc}(x)$ for any $x \in \{0,1\}^n$ **then**
  Set $\rho_q = (x \mapsto R(x))_{\text{Enc}}$
end
if q is a query of the form $\text{Dec}_{\mathsf{V}}(w, c)$ for any $w \in \{0,1\}^*$ and $c \in \{0,1\}^{2n}$ **then**
  if $\exists\ (x \mapsto c)_{\text{Enc}} \in U$ **then**
    Emulate $b \leftarrow \mathsf{V}^{\Theta}(w, x)$
    for each query q_{V} asked by V **do**
      $(\rho_{\mathsf{V}}, W_{\mathsf{V}}) \leftarrow \texttt{EmulateCall}^R(U \cup W_H, q_{\mathsf{V}})$
      $W_H = W_H \cup (\rho_{\mathsf{V}} \cup W_{\mathsf{V}})$
    end
    if $b = 1$ **then**
      Set $\rho_q = ((w, c) \mapsto x)_{\text{Dec}}$ /* Canonical caller can now call $\mathsf{V}^{\Theta}(w, x)$ to publicize W_H */
    else
      Set $\rho_q = ((w, c) \mapsto \perp)_{\text{Dec}}$
    end
  else
    Set $\rho_q = ((w, c) \mapsto \perp)_{\text{Dec}}$
  end
end
Return (ρ_q, W_H)

6.4.1 The New Obfuscator $\widehat{O}^R$ in the Random Oracle Model

Let $R = \{R_n\}_{n \in N}$ 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 rIHWE 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.* Given input circuit $C \in \mathcal{C}_n$, emulate $iO^{\Theta}(C)$ and let B be the output of this emulation. Let T_O be the transcript of this phase and initialize $Q_O := U(T_O) = \varnothing$. For

every query q asked by $iO^{\Theta}(C)$:

- Call $(\rho_q, W_H) \leftarrow \texttt{EmulateCall}^R(Q_O, q)$ as in Algorithm 2 and return the answer in ρ_q
- Add ρ_q and W_H to Q_O

Note that we will not pass *any* query-answer pairs Q_O asked by iO^{Θ} to the output of this new obfuscation. All of the query-answer pairs in Q_O, be it visible or hidden, will only be used for emulation purposes in the following step.

2. *Learning phase.* Let Q_B be the set of all visible queries that will be learned, Q_B^h be the set of hidden queries that we will keep track of for emulation purposes. Set $m = 3\ell_O/\epsilon$ where $\ell_O \leq |iO|$ represents the number of queries asked by iO. Choose $t \xleftarrow{\$} [m]$ uniformly at random then for $i = \{1, ..., t\}$ do the following:

   - Choose $z_i \xleftarrow{\$} \{0,1\}^{|C|}$ uniformly at random
   - Run $Ev^{\Theta}(B, z_i)$. For every query q asked by $Ev^{\Theta}(B, z_i)$:
     - Call $(\rho_q, W_H) \leftarrow \texttt{EmulateCall}^R(Q_O \cup Q_B \cup Q_B^h, q)$ and return the answer in ρ_q
     - Add ρ_q to Q_B and add W_H to Q_B^h.

   Since Ev is a canonical algorithm, in addition to the direct queries asked by $Ev^{\Theta}(B, z_i)$ to Θ, the resulting query set Q_B would also include the indirect queries that have resulted from $Ev^{\Theta}(B, z_i)$ running $\mathsf{V}^{\Theta}(w, x) = 1$ on a successful decryption query.

3. The output of the random oracle model obfuscation algorithm $\widehat{iO}^R(C)$ will be $\widehat{B} = (B, Q_B)$ where Q_B is the set of visible (direct and/or indirect) query-answer pairs.

Subroutine $\widehat{Ev}^R(\widehat{B}, z)$. Initialize $Q_{\widehat{B}} = \varnothing$ to be the set of queries asked when evaluating $\widehat{B}$. To evaluate $\widehat{B} = (B, Q_B)$ on a new random input z we simply emulate $Ev^{\Theta}(B, z)$. For every query q asked by $Ev^{\Theta}(B, z)$, run and set $(\rho_q, W_H) = \texttt{EmulateCall}^R(Q_B \cup Q_{\widehat{B}}, q)$ then return the answer in ρ_q and add (ρ_q, W_H) to $Q_{\widehat{B}}$.

Note that, since we are emulating query calls with respect to $Q_B \cup Q_{\widehat{B}}$, all the indirect query-answer pairs W_H returned from queries of the form $\text{Dec}_{\mathsf{V}}(w, c)$ are due to knowing the underlying $(x \mapsto c)_{\text{Enc}}$ (i.e. $(x \mapsto c)_{\text{Enc}} \in Q_B \cup Q_{\widehat{B}}$) so W_H is visible and therefore used in the emulation of any subsequent queries issued during this execution.

The running time of $\widehat{iO}$. 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 oracle where the PPT universal circuit evaluator V can have oracle gates to subroutines of Θ, we emphasize that in our framework of extended primitives, the effective running time of V, which we are executing during `EmulateCall`, remains to be a strict polynomial in the size of V and so the issue of exponential or infinite recursive calls is non-existent.

Proving approximate correctness. Let $Q_S = (Q_O \cup Q_B \cup Q_B^h \cup Q_{\widehat{B}})$ be the set of all query-answer pairs asked during the emulation, learning, and evaluation phases. We define the following hybrid experiments to show that the correctness of the new random-oracle model obfuscator that simulates an execution of a Θ-model obfuscation is sufficiently close to the correctness of an ideal execution of the Θ-model obfuscation. We denote the distribution of Hybrid i as $H^i = (Q_O^i, Q_B^i, Q_B^{i,h}, Q_{\widehat{B}}^i)$.

- Hybrid 0: This is the real experiment, which is represented in Section 6.4.1.
- Hybrid 1: This is the same as Hybrid 0 except for two differences:
  - The queries asked during emulation (Step 1) and learning (Step 2) answered relative to a true oracle Θ instead of emulating them using `EmulateCall`.
  - We modify the final execution $\widehat{Ev}(\widehat{B}, z)$ such that, when we execute $Ev^{\Theta}(B, z)$, we would forward all queries to Θ instead of using `EmulateCall` *except* when the query is of the form $\mathrm{Dec}(w, c)$ for some $(x \mapsto c)_{\mathrm{Enc}} \notin Q_B \cup Q_{\widehat{B}}$, in which case we *do not* forward this query to Θ and use $\perp$ as the answer instead. We will refer to such decryption queries as being *unknown*.
- Hybrid 2: This is the ideal experiment, where *all* the queries asked during emulation, learning and final execution are answered relative to a true oracle Θ. That is, all queries whose answers are emulated using `EmulateCall` throughout the obfuscation and final evaluation phases will instead be answered directly using Θ.

Claim 6.13. $\Delta(H^0, H^1) \leq \mathrm{negl}(\kappa)$

Proof. We first observe that the emulation and learning processes of Hybrid 0 is a statistical simulation of the true oracle. Furthermore, if the final execution phase asks a query $\mathrm{Dec}(w, c)$ for which $(x \mapsto c)_{\mathrm{Enc}} \in Q_B \cup Q_{\widehat{B}}$ then the answer in both hybrids will be consistent with each other otherwise the answer is $\perp$ in both hybrids. Consequently, any encryption query will be answered in Hybrid 0 using random oracle R and in Hybrid 1 using Θ, and Hybrid 0 statistically simulates decryptions queries without access to Θ.

The only event that can cause a discrepancy between these two hybrids is during the emulation and learning phases where a decryption query $\mathrm{Dec}_{\mathsf{V}}(w, c)$ is issued and $(x \mapsto c)_{\mathrm{Enc}} \notin Q_S$ but c is a valid ciphertext (i.e. there exists x such that $\mathrm{Enc}(x) = c$). In that case, in Hybrid 0, the answer to such a query will be $\perp$ whereas the answer in Hybrid 1 will be x. However, the probability of that happening in Hybrid 1 (over the randomness of oracle Θ) is at most $(2^n - p)/(2^{2n} - p) \leq 1/2^{n-1} = \mathrm{negl}(\kappa)$ given that $p = tl_B l_O = \mathrm{poly}(\kappa)$. □

Claim 6.14. $\Delta(H^1, H^2) \leq \epsilon$

Proof. Note that, since the emulation and learning processes are the same in both hybrids, we have that $(Q_O^1, Q_B^1, Q_B^{1,h}) = (Q_O^2, Q_B^2, Q_B^{2,h})$ and so we omit the superscripts when referencing these query-answer sets for simplicity of notation. In Hybrid 1, the execution of $\widehat{B}$ emulates $Ev^{\Theta}(B, z)$ on a random input z and answers all queries using Θ except for unknown decryption queries, whereas in Hybrid 2, we execute $\widehat{B}$ and answer *all* of $Ev^{\Theta}(B, z)$'s queries using the actual oracle Θ. In essence, in Hybrid 1, we can think of the execution as $Ev^{\widehat{\Theta}}(B, z)$ where $\widehat{\Theta}$ is the oracle simulated by

$\widehat{B}$ using Q_B and Θ. We will identify the events that differentiate between the executions $Ev^{\Theta}(B,z)$ and $Ev^{\widehat{\Theta}}(B,z)$.

Assume that the query-answer pairs so far during the execution of $\widehat{B}$ are the same in both hybrids. That is, we start with some $Q^1_{\widehat{B}} = Q^2_{\widehat{B}} = Q_{\widehat{B}}$ for the sake of proving inductively that any subsequent query-answer pairs are closely distributed between the two hybrids. Let q be a new (possibly indirect) query that is being asked by $Ev^{\widehat{\Theta}}(B,z)$. We present a case-by-case analysis of all possible query types to identify the cases that can cause discrepancies between the two hybrids:

1. If q is a query of type $\mathrm{Enc}(x)$, then it will be answered the same in both hybrids using the true oracle Θ.

2. If q is of type $\mathrm{Dec}(w,c)$ for which there exists $(x \mapsto c)_{\mathrm{Enc}} \in Q_B \cup Q_{\widehat{B}}$ then the answer to q would be determined the same in both hybrids. In particular, both hybrids will run $\mathsf{V}^{\Theta}(w,c) = b$, adding any indirect queries to $Q_{\widehat{B}}$ and answering q with x if and only if $b = 1$.

3. If q is of type $\mathrm{Dec}(w,c)$ such that $(x \mapsto c)_{\mathrm{Enc}} \notin Q_S$ then this means that we are attempting to decrypt a ciphertext for which we have not encrypted before (either directly or indirectly). In Hybrid 2, since Enc is a sparse random injective function, the answer would be $\perp$ with overwhelming probability as c would be invalid, and in Hybrid 1 the answer will also be $\perp$ by the definition of $\widehat{Ev}(\widehat{B},z)$ in this hybrid. Note that in both hybrids, there will not be any indirect queries in $Q_{\widehat{B}}$ as a result of this query since the ciphertext c is deemed invalid and V will not even be executed.

4. Suppose q is of type $\mathrm{Dec}(w,c)$ that is not determined by $Q_B \cup Q_{\widehat{B}}$ in Hybrid 1 and yet is determined by Q_S in Hybrid 2. We list the different cases here that may cause problems:

   (a) If q is of type $\mathrm{Dec}(w,c)$ such that $(x \mapsto c)_{\mathrm{Enc}} \in Q_S \setminus (Q_B \cup Q_{\widehat{B}})$ and $\mathsf{V}^{\Theta}(w,x) = 0$ then this means that we are attempting to decrypt a ciphertext for which the decryption will fail. The answer will be $\perp$ in Hybrid 1 by the definition of $\widehat{Ev}(\widehat{B},z)$ and is also $\perp$ in Hybrid 2 since we are using the real oracle Θ. However, one crucial point here is that, while there will possibly exist hidden queries due to the internal execution of $\mathsf{V}^{\Theta}(w,x)$ in Hybrid 2, such hidden queries will *not* be present in Hybrid 1 as we do not run V there. As we shall see later on, this is a potential source of inconsistency for later queries.

   (b) **Bad event 1.** The query-answer pair $(x \mapsto c)_{\mathrm{Enc}}$ is in $Q_O \setminus (Q_B \cup Q_{\widehat{B}})$ and $\mathsf{V}^{\Theta}(w,x) = 1$. That is, we are for the first time decrypting a ciphertext that was encrypted in Step 1 because we failed to learn the answer of this decryption query during the learning phase of Step 2. In that case, in Hybrid 1, the answer would be $\perp$ since we do not know the corresponding message x whereas in Hybrid 2 it would use the answer consistent with $Q_O \subseteq Q_S$ and output x.

   (c) **Bad event 2.** The query-answer pair $(x \mapsto c)_{\mathrm{Enc}}$ is in $Q^h_B \setminus (Q_B \cup Q_{\widehat{B}})$ and $\mathsf{V}^{\Theta}(w,x) = 1$. That is, we are for the first time decrypting a ciphertext from a hidden encryption query in Step 2. In that case, in Hybrid 1, the answer would be $\perp$ since we do not know the corresponding message x whereas in Hybrid 2 it would use the answer consistent with $Q^h_B \subseteq Q_S$ and output x.

   (d) **Bad event 3.** The query-answer pair $(x \mapsto c)_{\mathrm{Enc}}$ is in $Q^h_{\widehat{B}} \setminus (Q_B \cup Q_{\widehat{B}})$ and $\mathsf{V}^{\Theta}(w,x) = 1$ where $Q^h_{\widehat{B}}$ is the set of hidden queries generated in Hybrid 2 as a result of queries from

Case 4a. In that case, in Hybrid 1, the answer would be $\perp$ since we do not know the corresponding message x whereas in Hybrid 2 it would use the answer consistent with $Q^h_{\widehat{B}} \subseteq Q_S$ and output x.

While the above bad events will generate indirect queries in Hybrid 2 (the ciphertext is valid there), these indirect queries will not be present in Hybrid 1, and may cause inconsistencies between the two hybrids when evaluating subsequent queries. However, as long as we show that the bad events happen with small probability, those indirect queries will also be generated in Hybrid 2 only with small probability.

For input x, let $E_1(x)$ be the event that Case 4b happens, $E_2(x)$ be the event that Case 4c happens, and $E_3(x)$ be the event that Case 4d happens. Assuming that event $E(x) = (E_1(x) \vee E_2(x) \vee E_3(x))$ does not happen, both experiments will proceed identically the same and the output distributions of $Ev^{\Theta}(B, x)$ and $Ev^{\widehat{\Theta}}(B, x)$ will be statistically close. More formally, the probability of correctness for $\widehat{O}$ 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 O, 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 $\Pr[E(x)] = \Pr[E_1(x)] + \Pr[E_2(x) \mid \overline{E}_1(x)] + \Pr[E_3(x) \mid \overline{E}_1(x) \wedge \overline{E}_2(x)] \leq \epsilon$. 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 these bad events in Hybrid 2.

Claim 6.15. $\Pr_x[E_1(x)] \leq \epsilon/3$.

Proof. Recall that E_1 is the event that during the final execution of the obfuscation $\widehat{B}$ on a random input, a $\mathrm{Dec}(w, c)$ query is asked for some c that was generated during the obfuscation phase but such a decryption query was never asked during the learning phase (Case 4b). For all $i \in [t]$, let $Q'_{B_i} = (Q_{B_i} \cup Q^h_{B_i}) \cap Q_O$ 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). 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 increase the size of the set of learned obfuscation queries is at most ℓ_O. 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 Hybrid 1, the real execution $Q'_{\widehat{B}} := Q'_{B_{t+1}}$ would contain a new unlearned query that was in Q_O. Thus, given that $m = 3\ell_O/\epsilon$, the probability (over the selection of t) that t is bad is at most $\ell_O/m < \epsilon/3$. □

Claim 6.16. $\Pr_x[E_2(x) \mid \overline{E}_1(x)] \leq \text{negl}(\kappa)$.

Proof. Recall that E_2 is the event that during the final execution of the obfuscation $\widehat{B}$ on a random input, a query is asked that was issued during the learning phase but was part of the hidden set and therefore never publicized (Case 4c).

For all $i \in [t]$, let $Q_{B_i} = (Q_{B_i} \cup Q^h_{B_i})$ be the set of (public and hidden) query-answer pairs generated by the i'th evaluation $Ev^{\Theta}(B, z_i)$ during the learning phase (Step 2) and $Q_{\widehat{B}}$ be all the query-answer pairs of the final evaluation of the obfuscation. Note that in Hybrid 2, the real oracle Θ is used and therefore each execution of $Ev^{\Theta}(B, z_i)$ is independent of the other executions. In particular, we can think of the final evaluation as iteration $t+1$ of the learning phase with query-answer pairs $Q_{B_{t+1}} = Q_{\widehat{B}}$. Now, we can think of an intermediate hybrid with distribution $H^{2'}$ where the executions are randomly permuted in a way such that execution $t+1$ in Hybrid 2 is now the first execution in Hybrid $2'$. Note that, since the executions are independent, the distributions H^2 and $H^{2'}$ are equivalent and it remains to show that E_2 is unlikely to happen in Hybrid $2'$. Specifically, we need to show that the first execution of the learning phase (with query-answer pairs $Q_{B_{t+1}}$) does not decrypt a ciphertext generated by any previous hidden queries in the learning phase. However, since this is the first execution, there exists no hidden queries from previous executions and it suffices to argue that it does not decrypt a ciphertext generated by any hidden queries $Q^h_{B_{t+1}}$ from its own execution.

Since E_1 does not happen and since the probability that $Ev^{\Theta}(B, z_{t+1})$ decrypts a ciphertext that was never encrypted during the whole process is negligible, the only ciphertexts that this execution will attempt to decrypt are ones that were generated during this very same execution. Therefore, E_2 only happens with negligible probability when a ciphertext that was never encrypted before is being decrypted.

□

Claim 6.17. $\Pr[E_3(x) \mid \overline{E}_1(x) \wedge \overline{E}_2(x)] \leq \text{negl}(\kappa)$.

Proof. We bound this event in Hybrid 1. Given that the final execution in this hybrid does not issue unknown decryption queries, there are no hidden queries generated in this hybrid and so E_3 only happens with negligible probability when a ciphertext that was never encrypted before is being decrypted. □

Combining the results of the above claims, we get that $\Pr_x[E(x)] \leq \Pr[E_1(x)] + \Pr[E_2(x) \mid \overline{E}_1(x)] + \Pr[E_3(x) \mid \overline{E}_1(x) \wedge \overline{E}_2(x)] \leq \epsilon/3 + \text{negl}(n) \leq \epsilon$.

□

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 $\widehat{O}$ 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 $\widehat{O}$ and, therefore, security follows.

□

7 Separating IO from Homomorphic Witness Encryption

In this section, we formally prove our third main separation theorem which states that there is no fully black-box construction of indistinguishability obfuscation from any *extended* fully homomorphic encryption scheme.

Theorem 7.1. *Assume the existence of one-way functions and that* $\mathbf{NP} \not\subseteq \mathbf{coAM}$. *Then there exists no monolithic construction of indistinguishability obfuscation (IO) from fully homomorphic encryption (FHE).*

In fact, we prove a stronger result by showing a separation of IO from a more generalized and powerful version of witness encryption, which we call extended *extractable homomorphic* witness encryption. In essence, this is an instance-revealing witness encryption (Definition 3.4) with added homomorphic capabilities and extractable security.

Definition 7.2 (Extended Extractable Instance-Revealing Homomorphic Witness Encryption)**.** Let V be a universal circuit-evaluator that takes instance a and witness w and either accepts or rejects. Furthermore, let F be a universal circuit evaluator that takes as input a sequence of messages $m_1, ..., m_k$ to output some value m'. For any given security parameter κ, an extended *extractable instance-revealing homomorphic witness encryption* (ex-EIRHWE) scheme defined for (V, F) consists of four PPT algorithms $P = (\mathrm{Enc}, \mathrm{Rev}, \mathrm{Dec}_{\mathsf{V}}, \mathrm{Eval}_{\mathsf{F}})$ 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 $a \in \{0,1\}^* \cup \{\perp\}$.
- $\mathrm{Dec}_{\mathsf{V}}(w, c)$: given ciphertext c and "witness" string w, it outputs a message $m' \in \{0,1\}^*$.
- $\mathrm{Eval}_{\mathsf{F}}(c_1, ..., c_k)$: given ciphertexts[18] $c_1 = \mathrm{Enc}(a_1, m_1, 1^\kappa), ..., c_k = \mathrm{Enc}(a_k, m_k, 1^\kappa)$, it outputs $\{c'_i\}_{i=1}^p$ where $p = |U|$ and $U \subseteq A$ is the set of distinct attributes from $A = \{a_1, ..., a_k\}$.

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

- **Decryption Correctness.** For any security parameter κ, we have that the following two conditions are satisfied:
  - For any (w, a) such that $\mathsf{V}^P(w, a) = 1$, and any m:
  $$\Pr_{\mathrm{Enc,Dec}}[\mathrm{Dec}(w, \mathrm{Enc}(a, m, 1^\kappa)) = m] = 1.$$
  - For any (w_i, a_i) such that $\mathsf{V}^P(w_i, a_i) = 1$, and any $m_1, ..., m_k$:
  $$\Pr_{\mathrm{Enc,Dec,Eval}}[\mathrm{Dec}(w_i, \mathrm{Eval}(c_1, ..., c_k)) = \mathsf{F}^P(m_1, ..., m_k)|_i] = 1$$
  where $c_i = \mathrm{Enc}(a_k, m_k, 1^\kappa)$ for all $i \in [k]$ and $\mathsf{F}^P(m_1, ..., m_k)|_i$ denotes the ith secret share of $\mathsf{F}^P(m_1, ..., m_k)$.

[18] Without loss of generality, we exclude from the input to the evaluation any explicit in-the-clear representation of a function $f \in \mathcal{F}$ to compute on the underlying messages as we can interpret the function f as a "message" $m_f = f$, encrypt it to get c_f and include it as input to the evaluation.

- **Instance-revealing correctness.** For security parameter κ and any (a, m) it holds that:

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

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

- **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 and any $m_0 \neq m_1$ of the same length $|m_0| = |m_1|$, if:

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

  then,

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

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

Theorem 7.3. *Assume the existence of one-way functions and that* $\mathbf{NP} \not\subseteq \mathbf{coAM}$*. Then there exists no* monolithic *construction of indistinguishability obfuscation from extractable instance-revealing homomorphic witness encryption for any PPT verification algorithm* V *and function* F*.*

Since extended EIRHWE implies extended fully homomorphic encryption (in fact, in Sections 8.2 and 8.3, we show that it also implies other more powerful primitives such as extended attribute-based FHE and spooky encryption), Theorem 7.1 follows from Theorem 7.3, the above observations, and Lemma 4.10 (the transitivity lemma). As a result, for the remainder of this section we will focus on proving Theorem 7.3.

7.1 Overview of Proof Techniques

To prove Theorem 7.3, we will apply Lemma 3.26 for the rIRHWE model Ψ (formally defined in Section 7.2) to prove that there is no black-box construction of IO from any primitive $\mathcal{P}$ that can be oracle-fixed constructed from the Ψ. In particular, we will do so for $\mathcal{P}$ that is the extended IRHWE primitive. Our task is thus twofold: **(1)** to prove that $\mathcal{P}$ can be oracle-fixed constructed from Ψ and **(2)** to show a simulatable compilation procedure that compiles out Ψ from any IO construction. The first task is proven in Section 7.3 and the second task is proven in Section 7.4. By Lemma 3.26, this would imply the separation result of IO from $\mathcal{P}$ and prove Theorem 7.3.

Our oracle, which is more formally defined in Section 7.2, resembles an idealized version of a homomorphic witness encryption scheme, which makes the construction of extended IRHWE straightforward. As a result, the main challenge lies in showing a simulatable compilation procedure for IO that satisfies Definition 3.23 in this idealized model, and therefore, it is instructive to look at how the compilation process works and what challenges are faced with dealing with oracle Ψ.

7.1.1 High Level Compiler Structure and Challenges

Here we briefly discuss the general structure of the proposed compiler before going over the issues that arise when dealing with how to compile out Ψ. In this ideal model, we will aim to compile out both the decryption and evaluation procedures to reduce the Ψ oracle into a basic random oracle. Note, however, that while we will be using an instance-revealing oracle (and therefore we can run V on the revealed attribute to discover its indirect queries similar to the IRWE case), we still have to handle those hidden queries issued by the evaluation function F.

Challenge faced with homomorphic witness encryption. Similarly to the case of compiling out rIHWE, since Ψ is an extended oracle, the algorithm F is allowed to issues queries to any subroutine in Ψ. As a result, during the learning process of the compilation procedure, a query to $\mathrm{Eval}_{\mathsf{F}}$ might be asked for which queries issued by F would remain hidden (see Definition 6.12) and therefore may affect correctness when a new execution of $\widehat{B}(x)$ hits one of those queries.

Resolving the challenge. We resolve this problem by canonicalizing the emulation and learning processes as well as the obfuscation B so that we force the obfuscation B, upon issuing a query q of the form $\mathrm{Eval}_{\mathsf{F}}(c_1, ..., c_k)$, to explicitly reveal any queries asked by F by having it call $\mathsf{F}^{\Psi}(m_1, ..., m_k)$ if it knows *all* of the encryptions $\mathrm{Enc}(a_i, m_i) = c_i$. However, we still need to consider the case that, when executing $\widehat{B}^{\Theta}(x)$, at least one of the encryptions are unknown, in which case, we simply emulate the answer of q to be set of uniformly *random* ciphertexts (which we will categorize as fake ciphertexts). We argue that unless a Dec query is asked to decrypt one of the fake ciphertexts then the real execution's distribution is close to an ideal execution's distribution. We prove that due to the learning procedure, such a decryption query has a low probability of happening.

7.2 The Ideal Model

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

Definition 7.4 (Random Instance-Revealing Witness Homomorphic Encryption Oracle)**.** Let V be a universal circuit-evaluator that takes instance a and witness w and either accepts or rejects. Furthermore, let F be a universal circuit evaluator that takes as input a sequence of messages $m_1, ..., m_k$ to output some value m'. Let $H : \{0,1\}^* \to \{0,1\}^{n/2}$ be a public random hash function. We define the following *random instance-revealing witness homomorphic encryption* (rIRWHE) oracle $\Psi_{\mathsf{V},\mathsf{F},n} = (\mathrm{Enc}, \mathrm{Rev}, \mathrm{Dec}_{\mathsf{V}}, \mathrm{Eval}_{\mathsf{F}})$ as follows:

- $\mathrm{Enc}\colon \{0,1\}^n \mapsto \{0,1\}^{2n}$ is a random injective function. We will use the Enc oracle to encrypt a $m \in \{0,1\}^{n/2}$ with respect to attribute $a \in \{0,1\}^{n/2}$ to obtain a ciphertext $c \in \{0,1\}^{2n}$.

  For any $c \in \{0,1\}^{2n}$, we call c valid if there exists x such that $\mathrm{Enc}(x) = c$ and fake otherwise.

- $\mathrm{Rev}\colon \{0,1\}^{2n} \mapsto \{0,1\}^{n/2} \cup \perp$ is a function that, given an input $c \in \{0,1\}^{2n}$, would output a for which $\mathrm{Enc}((a,m)) = c$. If there is no such preimage, then it outputs $\perp$ instead.

- $\mathrm{Dec}_{\mathsf{V}}\colon \{0,1\}^s \mapsto \{0,1\}^n \cup \{\perp\}$: Given $w \in \{0,1\}^k, c \in \{0,1\}^{2n}$ as inputs where $k = \mathrm{poly}(n)$ and $s = k + 2n$, $\mathrm{Dec}(w,c)$ 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 $\text{Enc}(x) = c$.
3. If $\mathsf{V}^{\Theta}(w, a) = 0$ output $\perp$. Otherwise, output $x = (a, m)$.

- Eval_{F}: $\{0,1\}^{2nk} \mapsto \{0,1\}^{2np}$: Given a sequence of inputs $c_1, ..., c_k$, this subroutine performs the following:

  1. If for some $i \in [k] \nexists\, x_i = (a_i, m_i)$ for $\text{Enc}(x_i) = c_i$, then output $\perp$. Otherwise, continue to the next step.
  2. For each $i \in k$, find $x_i = (a_i, m_i)$ such that $\text{Enc}(x_i) = c_i$ (this is an inefficient process). Let $\{a'_1, ..., a'_p\}$ be the set of distinct attributes embedded in $x_1, ..., x_k$ where $p \leq k$.
  3. Run $b = \mathsf{F}^{\Psi}(m_1, ..., m_k)$
  4. Output $\{c'_i\}_{i=1}^{p}$ where, for all $i \in [p]$, $c'_i = \text{Enc}(x'_i)$, $x'_i = (a'_i, b_i||h_i)$, where the value h is defined as $h_i = H(i, a_1, ..., a_k, m_1, ..., m_k)$ and b_i is the i'th share of b.

We define a query-answer pair resulting from query q to subroutine $T \in \{\text{Enc}, \text{Rev}, \text{Dec}_{\mathsf{V}}, \text{Eval}_{\mathsf{F}}\}$ with some answer β as $(q \mapsto \beta)_T$. For simplicity, when n is clear from the context, we use $\Psi = (\text{Enc}, \text{Rev}, \text{Dec}_{\mathsf{V}}, \text{Eval}_{\mathsf{F}})$ to refer to $\Psi_{\mathsf{V},\mathsf{F},n}$ for a fixed n.

Remark 7.5. Note that for the sake of achieving the most general case of extended primitives from this oracle, we will allow both F and V to have any oracle gates to $\Psi = (\text{Enc}, \text{Dec}_{\mathsf{V}}, \text{Rev}, \text{Eval}_{\mathsf{F}})$.

7.3 Homomorphic Witness Encryption Exists Relative to Ψ

In this section, we show how to construct a semantically-secure extended extractable IRHWE with corresponding universal-circuit evaluators V and F relative to $\Psi = (\text{Enc}, \text{Rev}, \text{Dec}_{\mathsf{V}}, \text{Eval}_{\mathsf{F}})$. More specifically, we will show how to construct a primitive (in the Ψ oracle model) that is simpler to prove the existence of and yet still implies EIRHWE.

Definition 7.6 (Extended Extractable One-way Homomorphic Witness Encryption (ex-EOHWE))**.** Let V be a universal circuit-evaluator that takes instance a and witness w and either accepts or rejects. Furthermore, let F be a universal circuit evaluator that takes as input a sequence of messages $m_1, ..., m_k$ to output some value m'. For any given security parameter κ, an *extended extractable one-way homomorphic witness encryption* scheme consists of the following PPT algorithms $P = (\text{Enc}, \text{Rev}, \text{Dec}_{\mathsf{V}}, \text{Eval}_{\mathsf{F}})$ defined as follows:

- $\text{Enc}(a, m, 1^{\kappa})$: given an instance $a \in \{0,1\}^*$, message $m \in \{0,1\}^*$, and security parameter κ (and randomness as needed) it outputs $c \in \{0,1\}^*$.
- $\text{Rev}(c)$: given ciphertext c returns the underlying attribute $a \in \{0,1\}^*$.
- $\text{Dec}_{\mathsf{V}}(w, c)$: given ciphertext c and "witness" string w, it outputs a message $m' \in \{0,1\}^*$.
- $\text{Eval}_{\mathsf{F}}(c_1, ..., c_k)$: given ciphertexts $c_1, ..., c_k$, outputs another sequence of ciphertexts $(c'_1, ..., c'_p)$ where $p \leq k$.

An extended extractable one-way homomorphic witness encryption scheme satisfies the same completeness properties of Definition 7.2 with the extractability property replaced with the following:

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

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

then,

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

Construction 7.7 (Extended Extractable One-Way Homomorphic Witness Encryption)**.** Let V be a universal circuit-evaluator that takes instance a and witness w and either accepts or rejects. Furthermore, let F be a universal circuit evaluator that takes as input a sequence of messages $m_1, ..., m_k$ to output some value m'. For any security parameter κ and oracle Ψ_κ sampled according to Definition 7.4, we will implement an extended EOHWE scheme P for messages $m \in \{0,1\}^k$ where $k = \text{poly}(\kappa)$ as follows:

- $\text{WEnc}(a, m, 1^\kappa)$: Given $a \in \{0,1\}^*$, message $m \in \{0,1\}^k$ and security parameter 1^κ, 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 $\perp$.
- $\text{WEval}(c_1, ..., c_k)$: Given ciphertexts $c_1, ..., c_k$, it outputs $\text{Eval}_{\mathsf{F}}(c_1, ..., c_k)$.

Remark 7.8 (From one-wayness to indistinguishability.)**.** We note that the primitive ex-EOHWE, which has one-way security, can be used to build an indistinguishability-based ex-IRHWE. For any a, since $\text{Enc}(a, .)$ is a random injective function (and hence one-way) we have that, by the Goldreich-Levin thoerem [GL89], there exists a hardcore predicate $b = \langle r, r' \rangle$ for the one-way function $\text{Enc}'(a, r, r') := (\text{Enc}(a, r), r')$. Now, to encrypt a one-bit message b under some attribute a, we would output the ciphertext $c = (\text{Enc}(a, r), r')$ where $r, r' \leftarrow \{0,1\}^k$ are randomly sampled conditioned on $b = \langle r, r' \rangle$. Furthermore, to perform an evaluation $\text{Eval}'_{\mathsf{F}'}(c_1, ..., c_p)$ over a set of ciphertexts of the form $c_i = (c'_i, r'_i)$ where $c'_i = \text{Enc}(a, r_i)$, we will first choose r'_{F} uniformly at random then run $\text{Eval}_{\mathsf{F}}(c'_i, ..., c'_p)$ where F is defined such that it outputs random r_{F} for which $\langle r_{\mathsf{F}}, r'_{\mathsf{F}} \rangle = \mathsf{F}'(\langle r_1, r'_1 \rangle, ..., \langle r_p, r'_p \rangle)$. We then output $c_{\mathsf{F}} = (\text{Enc}(a, r_{\mathsf{F}}), r'_{\mathsf{F}})$ as the new evaluated ciphertext.

Lemma 7.9. *Construction 7.7 is a correct and subexponentially-secure oracle-fixed implementation (Definition 3.16) of an extended extractable one-way homomorphic witness encryption in the ideal Ψ oracle model.*

To prove the security of this construction, we will show that if there exists an adversary A against scheme P (in the Ψ oracle model) that can invert an encryption of a random message with non-negligible advantage then there exists a (fixed) deterministic straight-line extractor E with access to $\Psi = (\text{Enc}, \text{Rev}, \text{Dec}_{\mathsf{V}}, \text{Eval}_{\mathsf{F}})$ 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 $m \xleftarrow{\$} \{0,1\}$ uniformly at random then runs $A^\Psi(1^\kappa, c^*)$ where $c^* \leftarrow \text{WEnc}(a, m, 1^\kappa)$ is the challenge. Let Q_A be the set of query-answer pairs for the encryption queries that A will ask. Queries issued by A are handled by E as follows:

- To answer any query $\text{Enc}(x)$ asked by A, it forwards the query to the oracle Ψ and returns some answer c. Add $(x \mapsto c)_{\text{Enc}}$ to Q_A.
- To answer any query $\text{Rev}(c)$ asked by A, it forwards the query to the oracle Ψ and returns some answer a.
- To answer any query $\text{Dec}_{\mathsf{V}}(w, c)$ asked by A, the extractor first issues a query $\text{Rev}(c)$ to get some answer a. If $a \neq \perp$, it would execute $\mathsf{V}^{\Psi}(w, a)$, forwarding queries asked by V to Ψ similar to how it does for A and adds any resulting query-answer pairs to Q_A. Finally, it forwards the query $\text{Dec}_{\mathsf{V}}(w, c)$ to Ψ to get some answer x. If $a = \perp$, it returns $\perp$ to A otherwise it returns x and adds $(x \mapsto c)_{\text{Enc}}$ to Q_A if $x \neq \perp$.
- To answer any query $\text{Eval}_{\mathsf{F}}(c_1, ..., c_k)$ asked by A, the extractor first issues queries $\text{Rev}(c_1), ...,$ $\text{Rev}(c_k)$ to get a sequence of answers $(a_1, ..., a_k)$. Let $(a'_1, ..., a'_p)$ be the set of unique attributes from $(a_1, ..., a_k)$. If $a'_i \neq \perp$ for all $i \in [p]$, it would check if there exists $((a_i, m_i) \mapsto c_i)_{\text{Enc}}$ for all $i \in [k]$ in Q_A. If so, it proceeds to run $m' \leftarrow \mathsf{F}^{\Psi}(m_1, ..., m_k)$, forwarding queries asked by F to Ψ, then finally returns $\{c'_i\}_{i=1}^{p}$ where $c'_i = \text{Enc}(a'_i, b_i | h_i)$, b_i is the i'th random additive share of m' and $h_i \leftarrow H(i, a_1, ..., a_k, m_1, ..., m_k)$. Finally, it adds all of $(x'_i \mapsto c'_i)_{\text{Enc}}$ to Q_A where $x'_i = (a'_i, b_i | h_i)$.

  However, if $(x_i \mapsto c_i)_{\text{Enc}} \notin Q_A$ for some $i \in [k]$, and in particular if $c_i = c^*$ for some i, then the extractor would instead emulate the ciphertext evaluations c'_i by encrypting a random message for each different attribute a'_i (since it does not know the underlying plaintext and cannot run F) and we call c'_i fake ciphertexts. Note that by the collision resistance of H, A would not be able to distinguish between such fake ciphertexts and real evaluated ones.

While handling the queries made by A, if a decryption query $\text{Dec}_{\mathsf{V}}(w, c)$ is issued by A (perhaps even indirectly as a result of running V or F) for some c such that $\text{Rev}(c) = \text{Rev}(c^*)$ (this includes either decrypting c^* directly or another c' having the same attribute as c^* that was perhaps generated as a result of an evaluation query with c^* as one of the inputs), the extractor will pass this query to Ψ, and if the result of the decryption is $x \neq \perp$ then the extractor will halt execution and output w as the witness for instance $x^* = (a, m)$. Otherwise, if after completing the execution of A, no such query was asked then the extractor outputs $\perp$. We prove the following lemma.

Lemma 7.10. *For any PPT adversary A and instance a, if there exists a non-negligible function $\epsilon(.)$ such that:*

$$\Pr\left[A^{\Theta}(1^{\kappa}, c) = m \mid m \xleftarrow{\$} \{0,1\}^{k}, c \leftarrow \text{WEnc}(a, m, 1^{\kappa})\right] \geq \epsilon(\kappa) \tag{5}$$

then there exists a PPT extractor E such that:

$$\Pr\left[E^{\Theta, A}(a) = w \wedge \mathsf{V}^{\Theta}(w, a) = 1\right] \geq \epsilon(\kappa) - \text{negl}(\kappa). \tag{6}$$

Proof. Let A be an adversary satisfying Equation (5) above and let AdvWin be the event that A succeeds in the inversion game. Furthermore, let ExtWin be the event that the extractor succeeds in extracting a witness (as in Equation (6) above). Note that, by our construction of extractor E, when E executes A, it perfectly simulates the oracle Ψ for A assuming that several bad events do not happen. Let B_1 be the bad event that A asks $\text{Dec}(w, c_1)$ or $\text{Eval}(c_1, ..., c_k)$ where $(x_i \mapsto c_i)_{\text{Enc}} \notin Q_A$ for some $i \in [k]$. Let B_2 be the bad event that A asks $\text{Dec}(w, c_1)$ or $\text{Eval}(c_1, ..., c_k)$

where $(x_i \mapsto c_i)_{\text{Enc}} \notin Q_A$ for some i is a hidden query that was asked due to a previous query $\text{Eval}_{\mathsf{F}}(y_1, ..., y_k)$ and $y_j = c^*$ or is otherwise fake for some j (so F could not be executed by the extractor). As long as $B = (B_1 \vee B_2)$ does not happens then the extractor will perfectly simulate A's view.

Note that for the special case of FHE, as long as B does not happen then all fake ciphertexts have the same attribute as c^* since the only case that $\text{Eval}(c_1, ..., c_k)$ would generate a fake ciphertext is when $\text{Rev}(c_i) = \text{Rev}(c^*)$ for all i. If A issues $\text{Dec}_{\mathsf{V}}(w, c'_i)$ for some fake ciphertext c'_i then the extractor would return the answer returned by Ψ even though that answer might be a random incorrect answer. However, this is safe to do since once A successfully decrypts a fake ciphertext (which must have an attribute equal to the attribute of c^*), the extractor wins immediately.

For the more general case where a fake ciphertext c'_i generated by $(c'_1, ..., c'_k) \leftarrow \text{Eval}(c_1, ..., c_k)$ queries may have different attributes (depending on the attribute value of the input ciphertexts), we argue that decrypting c'_i where $\text{Rev}(c'_i) \neq \text{Rev}(c^*)$ will not affect the distribution of the view of A. Recall that, by the design of the extractor, c'_i would be an encryption of a random message when in the ideal world it would have been an encryption of the i'th random additive share of $\mathsf{F}(m_1, ..., m_k)$. However, by the security of the random secret sharing, we can argue that, without knowing all the other secret shares, the distributions $\text{Dec}(w, c'_i)$ and $U_{n/2}$ are statistically close.

As a result of the above, the event B reduces to finding an image of a sparse random injective function, which is negligibly hard to accomplish. Now we show that whenever A wins then E must win as well. Observe that:

$$\Pr_{\Theta,m}[\overline{\mathsf{ExtWin}}] \leq \Pr_{\Theta,m}[\overline{\mathsf{ExtWin}} \wedge \mathsf{AdvWin} \wedge \overline{B}] + \Pr_{\Theta,m}[\overline{\mathsf{AdvWin}}] + \Pr_{\Theta,m}[B].$$

Since $\Pr[\mathsf{AdvWin}] \geq \epsilon$ for some non-negligible function ϵ and $\Pr[B] \leq \text{negl}(\kappa)$, it suffices to show that $\Pr[\overline{\mathsf{ExtWin}} \wedge \mathsf{AdvWin} \wedge \overline{B}]$ is negligible. Note that this event is equivalent to saying that the adversary succeeds in the inversion game but never asks a query of the form $\text{Dec}_{\mathsf{V}}(w, c')$ such that $\text{Rev}(c') = \text{Rev}(c^*)$ and for which the answer is $x \neq \perp$ so the extractor fails to recover the witness. For simplicity of notation define $\mathsf{Win} := \overline{\mathsf{ExtWin}} \wedge \mathsf{AdvWin} \wedge \overline{B}$.

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

Claim 7.11. *For any negligible δ,* $\Pr_{\Theta}\left[\Pr_m[\mathsf{Win}] \geq \sqrt{\delta}\right] \leq \sqrt{\delta}$.

Proof. We list all possible queries that A could ask and argue that these queries do not help A in any way without also forcing the extractor to win as well. Specifically, we show that for any such A that satisfies the event Win, there exists another adversary $\widehat{A}$ that depends on A and also satisfies the same event but does not ask any decryption or evaluation queries (only encryption queries). This would then reduce to the standard case of inverting a random injective function, which is known to be hard over the randomness of the oracle. We define the adversary $\widehat{A}$ as follows. Upon executing A, it handles the queries issued by A as follows:

- If A asks a query of the form $\text{Enc}(x)$ then $\widehat{A}$ forwards the query to Ψ to get the answer.
- If A asks a query of the form $\text{Rev}(c)$ then, since B does not happen, it must be the case that $((a, m) \mapsto c)_{\text{Enc}} \in Q_A$ and therefore $\widehat{A}$ returns a.

- If A asks a query of the form $\mathrm{Dec}(w, c^*)$ or $\mathrm{Dec}(w, c')$ where c' has the same attribute as c^* then w must be a string for which $\mathsf{V}^{\Psi}(w, a^*) = 0$ 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 $\mathsf{V}^{\Psi}(w, a^*)$ and answering its queries appropriately.

- If A asks a query of the form $\mathrm{Dec}(w, c')$ for some $c' \neq c^*$ then, since B 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 $\mathsf{V}^{\Psi}(w, a')$ and return the appropriate answer (x or $\perp$) depending on the answer of V.

- If A asks a query of the form $\mathrm{Eval}_{\mathsf{F}}(c_1, ..., c_k)$ and $((a_i, m_i) \mapsto c_i)_{\mathrm{Enc}} \in Q_A$ for all i then $\widehat{A}$ can compute $\beta = F^{\Psi}(m_1, ..., m_k)$, handling its queries, returning the correct encryptions of each share of β, and adding the resulting new encryptions to Q_A.

- If A asks a query of the form $\mathrm{Eval}_{\mathsf{F}}(c_1, ..., c_k)$ and $c_i = c^*$ or is otherwise a fake ciphertext then $\widehat{A}$ returns encryptions of random messages under the same attributes as the input ciphertexts (without running F). Note that, while an ideal execution would generate hidden encryption queries from running F, since B does not happen, these hidden encryption queries will not be decrypted with high probability.

Given that $\widehat{A}$ perfectly emulates A's view, the only possibility that A could win the inversion game is by asking $\mathrm{Enc}(x^*) = c^*$ and hitting the challenge ciphertext. By a standard averaging argument, we find that since $\Pr_{\Theta,m}[\mathsf{Win}] \leq \delta(\kappa)$ for some negligible δ then $\Pr_{\Theta}[\Pr_m[\mathsf{Win}] \leq \sqrt{\delta}] \geq 1 - \sqrt{\delta}$, which yields the result.

□

To conclude the proof of Lemma 7.10, we can see that the probability that the extractor wins is given by $\Pr[\mathsf{ExtWin}] \geq 1 - \Pr[\overline{\mathsf{ExtWin}} \wedge \mathsf{AdvWin} \wedge \overline{B}] - \Pr[\overline{\mathsf{AdvWin}}] - \Pr[B] \geq \epsilon(\kappa) - \mathrm{negl}(\kappa)$ where ϵ is the advantage to A.

□

Proof of Lemma 7.9. It is clear that the Construction 7.7 is correct. Furthermore, by Lemma 7.10, it also satisfies the extractability property. □

7.4 Compiling out Ψ from IO

In this section, we show a simulatable compiler for compiling out Ψ. We formalize the approach outlined in Section 7.1 while making use of Lemma 3.25, which allows us to compile out Ψ in two phases: we first compile out part of Ψ to get an approximately-correct obfuscator $\widehat{O}^R$ in the random oracle model, and then use the result of [CKP15] to compile out R and get an obfuscator $\widetilde{O}$ in the plain-model. Since we are applying this lemma only a constant number of times (in fact, just twice), security should still be preserved. Specifically, we will prove the following claim:

Lemma 7.12. *Let $R \sqsubset \Psi$ be a random oracle where "$\sqsubset$" denotes a sub-model relationship (see Definition 3.22). Then the following holds:*

- *For any IO in the Ψ ideal model, there exists a simulatable compiler with correctness error $\epsilon < 1/200$ for it that outputs a new obfuscator in the random oracle R model.*

- *For any IO in the random oracle model, there exists a simulatable compiler with correctness error $\epsilon < 1/200$ for it that outputs a new obfuscator in the plain model.*

Proof. The second part of Lemma 7.12 follows directly from [CKP15], and thus we focus on proving the first part of the claim. Before we start describing the compilation process, we present the following definition of canonical executions that is a property of algorithms in this ideal model and dependent on the oracle being removed.

Definition 7.13 (Canonical Executions)**.** We define an oracle algorithm A^{Ψ} relative to rIRHWE to be in canonical form if it satisfies all of the following requirements:

- Before asking any $\mathrm{Dec}_{\mathsf{V}}(w, c)$ query for a valid c query, A would first get $a \leftarrow \mathrm{Rev}(c)$ then run $\mathsf{V}^{\Psi}(w, a)$ on its own, making sure to answer any queries of V using Ψ.
- After asking a query $\mathrm{Dec}_{\mathsf{V}}(w, c)$ for which the returned answer is some message $m \neq \perp$, issue a query $a \leftarrow \mathrm{Rev}(c)$ followed by $\mathrm{Enc}(x)$ where $x = (a, m)$.
- Before A asks a query $\mathrm{Eval}_{\mathsf{F}}(c_1, ..., c_k)$ for which it knows the corresponding encryptions $c_1 = \mathrm{Enc}(a_1, m_1), ..., c_k = \mathrm{Enc}(a_k, m_k)$, it would first call $b \leftarrow \mathsf{F}^{\Psi}(m_1, ..., m_k)$, making sure to answer any queries of F using Ψ. Let $(a'_1, ..., a'_p)$ be the distinct attributes from $(a_1, ..., a_k)$ where $p \leq k$ and b_i be the i'th share of b where $i \in [p]$. A would then issue calls to $\mathrm{Enc}(x'_i)$ where for all $i \in [p], x'_i = (a_i, b_i | H(i, a_1, ..., a_k, m_1, ..., m_k))$.
- After A asks a query $\mathrm{Eval}_{\mathsf{F}}(c_1, ..., c_k)$ for which the returned answer is a sequence of ciphertexts $c'_1, ..., c'_p$ where $p \leq k$, issue queries $\mathrm{Rev}(c'_i)$ for all $i \in [p]$.

Note that any oracle algorithm A can be easily modified into a canonical form by increasing its query complexity by at most a polynomial factor (since V and F are PPT algorithms).

We now proceed to present the construction of the random oracle model obfuscator $O = (iO, Ev)$ that, given an obfuscator in the Ψ model, would compile out and emulate queries to Rev, Dec and Eval while forwarding any Enc queries to R. Throughout this process, we assume that the obfuscator iO and evaluation procedure Ev are all canonicalized according to Definition 7.13. For these canonical algorithms, let $l_O, l_B = \mathrm{poly}(\kappa)$, respectively, be the number of queries asked by the ideal-model obfuscator iO and the evaluation Ev to the rIRHWE oracle Ψ.

7.4.1 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 rIRWHE 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*: Given input $C \in \mathcal{C}_n$, emulate $iO^{\Psi}(C)$ and let B be the output of this emulation. Let T_O be the transcript of this phase and initialize $Q_O := Q(T_O) = \varnothing$. For every query q asked by $iO^{\Psi}(C)$:
   - Call $(\rho_q, W_H) \leftarrow \mathtt{EmulateCall}^{\Psi}(Q_O, q)$ as in Algorithm 3 and return the answer in ρ_q.

Algorithm 3: `EmulateCall`

Input: Query-answer set Q, query q
Oracle: Random Oracle R, Hash function H
Output: (ρ_q, W_H) where ρ_q is a query-answer pair containing the answer of query q and W_H is the set of indirect query-answer pairs (those added by F)

Begin.
Initialize $W_H = \varnothing$
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 $\rho_q = (c, a)$
  else
    Set $\rho_q = (c, \perp)$
  end
end
if q is a query of the form $\mathrm{Dec}_{\mathsf{V}}(w, c)$ **then**
  if $\exists\ (x \mapsto c)_{\mathrm{Enc}} \in Q$ for some $x = (a, m)$ **then**
    Emulate $b \leftarrow \mathsf{V}^{\Psi}(w, a)$ while handling its queries using `EmulateCall`
    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
if q is a query of the form $\mathrm{Eval}_{\mathsf{F}}(c_1, ..., c_k)$ **then**
  For all $i \in [k]$, get $(\rho_{a_i}, W_a) \leftarrow \mathtt{EmulateCall}(Q, \mathrm{Rev}(c_i))$ then add ρ_{a_i} to Q
  Let $\{a'_1, ..., a'_p\}$ be the distinct attributes of $\{a_1, ..., a_k\}$
  If $a_i = \perp$ for some i then return $((q \mapsto \perp)_{\mathrm{Eval}}, W_H)$
  if $\forall\ i\ \exists\ ((a_i, m_i) \mapsto c_i)_{\mathrm{Enc}} \in Q$ and c_i are valid **then**
    Emulate $b \leftarrow \mathsf{F}^{\Psi}(m_1, ..., m_k)$
    for each query q_{F} asked by F **do**
      $(\rho_{\mathsf{F}}, W_{\mathsf{F}}) \leftarrow \mathtt{EmulateCall}^{R}(Q \cup W_H, q_{\mathsf{F}})$
      $W_H = W_H \cup (\rho_{\mathsf{F}} \cup W_{\mathsf{F}})$
    end
    For all $a'_i \in \{a'_1, ..., a'_p\}$, set $c'_i \leftarrow \mathrm{Enc}(a'_i, b_i | H(i, a_1, ..., a_k, m_1, ..., m_k))$ and add to W_H
  else
    For all $a'_i \in \{a'_1, ..., a'_p\}$, set $c'_i \leftarrow \mathrm{Enc}(a'_i, r_i)$ where $r_i \xleftarrow{\$} \{0,1\}^{n/2}$ and add to W_H
  end
  Set $\rho_q = ((c_1, ..., c_k) \mapsto (c'_1, ..., c'_p))_{\mathrm{Eval}}$
end
Return (ρ_q, W_H)

- Add ρ_q and W_H to Q_O.

Note that we will not pass any query-answer pairs Q_O asked by iO^{Ψ} to the output of this new obfuscation. All of the query-answer pairs in Q_O will only be used for emulation purposes in the following step.

2. *Learning phase*: Let Q_B be the set of all visible queries that will be learned (which will be passed on to the obfuscated circuit) and let Q_B^h be the set of hidden queries that we will keep track of for emulation purposes (which will not be made public to the obfuscated circuit). Set $m = 4\ell_O/\epsilon$ where $\ell_O \leq |iO|$ represents the number of queries asked by iO. Choose $t \xleftarrow{\$} [m]$ uniformly at random then for $i = 1, ..., t$ do the following:

   - Choose $z_i \xleftarrow{\$} \{0,1\}^{|C|}$ uniformly at random
   - Run $Ev^{\Psi}(B, z_i)$. For every query q asked by $Ev^{\Psi}(B, z_i)$:
     - Call $(\rho_q, W_H) \leftarrow \texttt{EmulateCall}^R(Q_O \cup Q_B \cup Q_B^h, q)$ and return the answer in ρ_q
     - Add ρ_q to Q_B and add W_H to Q_B^h.

   Since Ev is a canonical algorithm, in addition to the direct queries asked by $Ev^{\Psi}(B, z_i)$ to Ψ, the resulting query set Q_B would also include the indirect queries that have resulted from $Ev^{\Psi}(B, z_i)$ running $\mathsf{V}^{\Psi}(w, a)$. Furthermore, if a query $\text{Eval}_{\mathsf{F}}(c_1, ..., c_k)$ is issued where $\text{Enc}((a_i, m_i)) = c_i$ then the indirect queries asked by $\mathsf{F}^{\Psi}(m_1, ..., m_k)$ will be made visible to the final execution so long as the query-answer pairs $((a_i, m_i) \mapsto c_i)_{\text{Enc}}$ for all i were directly asked beforehand (i.e. were in Q_B).

3. The output of the random oracle model obfuscation algorithm $\widehat{iO}^R(C)$ will be $\widehat{B} = (B, Q_B)$ where Q_B is the set of visible (direct/indirect) query-answer pairs.

Subroutine $\widehat{Ev}^R(\widehat{B}, z)$. Initialize $Q_{\widehat{B}} = \varnothing$ to be the set of queries asked when evaluating $\widehat{B}$. To evaluate $\widehat{B} = (B, Q_B)$ on a new random input z, we emulate $Ev^{\Psi}(B, z)$. For every query q asked by $Ev^{\Psi}(B, z)$:

- Run and set $(\rho_q, W_H) = \texttt{EmulateCall}^R(Q_B \cup Q_{\widehat{B}}, q)$ then return the answer in ρ_q.
- Add ρ_q and W_H to $Q_{\widehat{B}}$.

Remark 7.14 (Fake ciphertexts)**.** Throughout the circuit evaluation $\widehat{Ev}(\widehat{B}, z)$ subroutine, note that if a query of the form $\text{Eval}_{\mathsf{F}}(c_1, ..., c_k)$ is asked where at least one of the input ciphertexts c_i is not known to the evaluator, the answer to the query will be simulated as a sequence of encryptions of random messages (since at least one the underlying messages is unknown and therefore F cannot be computed). From here on, we refer to such simulated ciphertexts as *fake ciphertexts*.

The running time of $\widehat{iO}$. 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 oracle where the universal circuit evaluators V and F can have oracle gates to subroutines of Ψ, we emphasize that in our framework of extended primitives, the effective running times of V and F, which we are executing during `EmulateCall`, remain a strict polynomial in the size of the original circuits and so the issue of exponential or infinite recursive calls is non-existent.

Proving approximate correctness. Let $Q_S = (Q_O \cup Q_B \cup Q_B^h \cup Q_{\widehat{B}})$ be the set of all query-answer pairs asked during the emulation, learning, and final execution phases. We define the following hybrid experiments to show that the correctness of the new random-oracle model obfuscator that simulates an execution of a Ψ-model obfuscation is sufficiently close to the correctness of an ideal execution of the Ψ-model obfuscation. We denote the distribution of Hybrid i as $H^i = (Q_O^i, Q_B^i, Q_B^{i,h}, Q_{\widehat{B}}^i)$.

- Hybrid 0: This is the real experiment, which is represented in Section 7.4.1.
- Hybrid 1: This is the same as Hybrid 0 except for two differences:
  - The queries asked during emulation (Step 1) and learning (Step 2) answered relative to a true oracle Ψ instead of emulating them using `EmulateCall`.
  - We modify the final execution $\widehat{Ev}(\widehat{B}, z)$ such that, when we execute $Ev^{\Psi}(B, z)$, we would forward all queries to Ψ instead of using `EmulateCall` *except* when the query q is of the form $\text{Eval}(c_1, ..., c_k)$ where $(x_i \mapsto c_i)_{\text{Enc}} \notin Q_B \cup Q_{\widehat{B}}$ for some c_i, in which case we *do not* forward this query to Ψ and use the simulated answer returned by $\texttt{EmulateCall}(Q_B \cup Q_{\widehat{B}}, q)$ as the answer instead (which does not run F and returns fake ciphertexts instead).
- Hybrid 2: This is the ideal experiment, where *all* the queries asked during emulation, learning and final execution are answered relative to a true oracle Ψ. That is, all queries whose answers are emulated using `EmulateCall` throughout the obfuscation and final evaluation phases will instead be answered directly using Ψ.

Claim 7.15. $\Delta(H^0, H^1) \leq \frac{\epsilon}{2} + \text{negl}(\kappa)$.

Proof. We first observe that the emulation and learning processes of Hybrid 0 is a statistical simulation of the true oracle. The only event that can cause a discrepancy between these two hybrids during the emulation and learning phases is when a query of the form $\text{Rev}(c)$, $\text{Dec}_{\mathsf{V}}(w, c)$, or $\text{Eval}(c_1, ..., c_k)$ is issued but the emulator has never issued an encryption query that maps to an input of those queries (hitting a valid ciphertext without knowing the underlying plaintext). In that case, in Hybrid 0, the answer to such a query will be $\perp$ as defined by `EmulateCall` whereas the answer in Hybrid 1 will be consistent with the true oracle and return the correct answer. However, the probability of that happening in Hybrid 1 (over the randomness of oracle Ψ) is at most $(2^n - p)/(2^{2n} - p) \leq 1/2^{n-1} = \text{negl}(\kappa)$ given that $p = tl_B l_O = \text{poly}(\kappa)$.

Next, we need to show that the distribution of query-answer pairs in the final execution phase is close. In Hybrid 0, the execution of $\widehat{B}$ emulates $Ev^{\Psi}(B, z)$ on a random input z and emulates the answers of all queries using `EmulateCall`, whereas in Hybrid 1 we answer all queries of $Ev^{\Psi}(B, z)$ using the true Ψ except for $\text{Eval}(c_1, ..., c_k)$ queries for which $(x_i \mapsto c_i)_{\text{Enc}} \notin Q_B \cup Q_{\widehat{B}}$ for some c_i. In essence, in Hybrid 0, we can think of the execution as $Ev^{\widehat{\Psi}}(B, z)$ where $\widehat{\Psi}$ is the oracle simulated using just $Q_B \cup Q_{\widehat{B}}$, and in Hybrid 1, we can think of the execution as $Ev^{\widetilde{\Psi}}(B, z)$ where $\widetilde{\Psi}$ is the oracle simulated using just $Q_B \cup Q_{\widehat{B}}$ and Ψ except for evaluation queries of the aforementioned type (which generate fake ciphertexts). We will identify the events that differentiate between the executions $Ev^{\widehat{\Psi}}(B, z)$ and $Ev^{\widetilde{\Psi}}(B, z)$.

Assume that the query-answer pairs so far during the execution of $\widehat{B}$ are the same in both hybrids. That is, we start with some $Q_{\widehat{B}}^0 = Q_{\widehat{B}}^1 = Q_{\widehat{B}}$ for the sake of proving inductively that any

subsequent query-answer pairs are closely distributed between the two hybrids. Let q be a new (possibly indirect) query that is being asked by $Ev(B, z)$. We present a case-by-case analysis of all possible query types to identify the cases that can cause discrepancies between the two hybrids:

1. If q is a query of type $\text{Enc}(x)$, then it will be answered the same in both hybrids using the random oracle R in Hybrid 0 and using the true oracle Ψ in Hybrid 1.

2. If q is of type $\text{Rev}(c)$ for which there exists $((a, m) \mapsto c)_{\text{Enc}} \in Q_B \cup Q_{\widehat{B}}$ or $(c \mapsto a)_{\text{Rev}} \in Q_B \cup Q_{\widehat{B}}$ then the answer to q would be a in both hybrids.

3. If q is of type $\text{Dec}(w, c)$ for which there exists $(x \mapsto c)_{\text{Enc}} \in Q_B \cup Q_{\widehat{B}}$ then the answer to q would be determined the same (x is returned) in both hybrids.

4. If q is of type $\text{Eval}(c_1, ..., c_k)$ for which there exists $((a_i, m_i) \mapsto c_i)_{\text{Enc}} \in Q_B \cup Q_{\widehat{B}}$ then the answer to q would be determined the same in both hybrids. In particular, both hybrids will run $\mathsf{F}^{\Psi}(m_1, ..., m_k)$, adding any indirect queries to $Q_{\widehat{B}}$ and answering q with the a set of correctly generated ciphertexts.

5. If q is of type $\text{Rev}(c_1)$, $\text{Dec}(w, c_1)$, or $\text{Eval}(c_1, ..., c_k)$ such that there exists $(x_i \mapsto c_i)_{\text{Enc}} \notin Q_S$ for some $i \in [k]$ then this means that we are attempting to decrypt a ciphertext for which we have not encrypted before (either directly or indirectly). In Hybrid 1, since Enc is a sparse random injective function, the answer would be $\perp$ with overwhelming probability as c would be invalid, and in Hybrid 0 the answer will also be $\perp$ by the definition of $\widehat{Ev}(\widehat{B}, z)$ in this hybrid. Note that in both hybrids, there will not be any indirect queries in $Q_{\widehat{B}}$ as a result of this query since the ciphertexts that are being queried on are deemed invalid and neither V or F will be executed.

6. If q is of type $\text{Eval}(c_1, ..., c_k)$ such that there exists $(x_i \mapsto c_i)_{\text{Enc}} \in Q_S \setminus (Q_B \cup Q_{\widehat{B}})$ for some $i \in [k]$ then this means that we are attempting to decrypt a ciphertext that was not learned. However in both hybrids we will simulate the answer as encryptions of random messages and in both hybrids there will not be any hidden indirect queries in $Q_{\widehat{B}}$ as a result of this query.

7. Suppose q is of type $\text{Rev}(c)$ that is not determined by $Q_B \cup Q_{\widehat{B}}$ in Hybrid 0 and yet is determined by Q_S in Hybrid 1. We list the different cases here that may cause problems:

   (a) **Bad event 1.** The query-answer pair $((a, m) \mapsto c)_{\text{Enc}}$ is in $Q_O \setminus (Q_B \cup Q_{\widehat{B}})$. That is, we are for the first time revealing a valid ciphertext that was encrypted in Step 1 because we failed to learn the answer of this query during the learning phase of Step 2. In that case, in Hybrid 0, the answer would be $\perp$ since we do not know the corresponding attribute a whereas in Hybrid 1 it would use Ψ to produce the answer consistent with $Q_O \subseteq Q_S$ and output a.

   (b) **Bad event 2.** The query-answer pair $((a, m) \mapsto c)_{\text{Enc}}$ is in $Q_B^h \setminus (Q_B \cup Q_{\widehat{B}})$. That is, we are for the first time revealing a ciphertext from a hidden encryption query in Step 2. In that case, in Hybrid 0, the answer would be $\perp$ since we do not know the corresponding attribute a whereas in Hybrid 1 it would use the answer consistent with $Q_B^h \subseteq Q_S$ and output a.

8. Suppose q is of type $\text{Dec}(w, c)$ that is not determined by $Q_B \cup Q_{\widehat{B}}$ in Hybrid 0 and yet is determined by Q_S in Hybrid 1. We list the different cases here that may cause problems:

(a) If q is of type $\mathrm{Dec}(w, c)$ such that $((a, m) \mapsto c)_{\mathrm{Enc}} \in Q_S \setminus (Q_B \cup Q_{\widehat{B}})$ and $\mathsf{V}^{\Psi}(w, a) = 0$ then this means that we are attempting to decrypt a ciphertext for which the decryption will fail. The answer will be $\perp$ in Hybrid 0 by the definition of `EmulateCall` and is also $\perp$ in Hybrid 1 since we are using the real oracle Ψ. Note that in both hybrids, while there will be indirect queries (as a result of running $\mathsf{V}^{\Psi}(w, a)$), such queries will be visible in both worlds.

(b) **Bad event 3.** The query-answer pair $((a, m) \mapsto c)_{\mathrm{Enc}}$ is in $Q_O \setminus (Q_B \cup Q_{\widehat{B}})$ and $\mathsf{V}^{\Psi}(w, a) = 1$. That is, we are for the first time decrypting a valid ciphertext that was encrypted in Step 1 because we failed to learn the answer of this query during the learning phase of Step 2. In that case, in Hybrid 0, the answer would be $\perp$ since we do not know the corresponding message $x = (a, m)$ whereas in Hybrid 1 it would use Ψ to produce the answer consistent with $Q_O \subseteq Q_S$ and output x.

(c) **Bad event 4.** The query-answer pair $((a, m) \mapsto c)_{\mathrm{Enc}}$ is in $Q_B^h \setminus (Q_B \cup Q_{\widehat{B}})$ and $\mathsf{V}^{\Psi}(w, a) = 1$. That is, we are decrypting a ciphertext from a hidden encryption query in Step 2. In that case, in Hybrid 0, the answer would be $\perp$ since we do not know the corresponding message $x = (a, m)$ whereas in Hybrid 1 it would use the answer consistent with $Q_B^h \subseteq Q_S$ and output x.

(d) **Bad event 5.** The query-answer pair $\rho_e = ((c_1, ..., c_k) \mapsto (c'_1, ..., c'_p))_{\mathrm{Eval}}$ is in $Q_S \setminus Q_{\widehat{B}}$, $c = c'_i$ for some $i \in [p]$, $\exists (x_j \mapsto c_j)_{\mathrm{Enc}} \notin Q_B \cup Q_{\widehat{B}}$ and $\mathsf{V}^{\Psi}(w, a'_i) = 1$ where $a'_i = \mathrm{Rev}(c'_i)$. That is, we are decrypting one of the (non-fake) ciphertexts created from an evaluation query for which the underlying plaintexts of at least one of the ciphertext inputs to the evaluation are unknown to the evaluator. In that case, in Hybrid 0, the answer would be $\perp$ since we do not know the corresponding message underlying c whereas in Hybrid 1 it would use the answer supplied by Ψ and output the correct answer. We argue that that this bad event can be reduced to the previous bad events.

- If $\rho_e \in Q_O$ then this implies that $(x \mapsto c'_i)_{\mathrm{Enc}} \in Q_O$ due to our canonicalization procedure. As a result, this means that we are for the first time successfully decrypting a ciphertext from Step 1, which reduces to Case 8b.
- If $\rho_e \in Q_B \cup Q_B^h$ then this implies that $(x \mapsto c'_i)_{\mathrm{Enc}} \in Q_B^h$ due to our emulation procedure. As a result, this means that we are for the first time successfully decrypting a ciphertext from a hidden encryption query in Step 2, which reduces to Case 8c.

(e) The query-answer pair $((c_1, ..., c_k) \mapsto (c'_1, ..., c'_p))_{\mathrm{Eval}}$ is in $Q_{\widehat{B}}$, $c = c'_i$ for some $i \in [p]$, $\exists (x_j \mapsto c_j)_{\mathrm{Enc}} \notin Q_B \cup Q_{\widehat{B}}$ and $\mathsf{V}^{\Psi}(w, a'_i) = 1$ where $a'_i = \mathrm{Rev}(c'_i)$. That is, we are decrypting a fake ciphertext created from an evaluation query for which the underlying plaintexts of at least one of the ciphertext inputs to the evaluation are unknown to the evaluator. However, since the query was generated during the evaluation phase, the fake ciphertext c was also generated during the evaluation and therefore $(x \mapsto c)_{\mathrm{Enc}}$ is known where $x = (a||r)$ for some random r. Thus, in both Hybrids, the answer would be x.

For input x, let $E_1(x)$ be the event that Case 8b or Case 7a happens, and $E_2(x)$ be the event that Case 8c or Case 7b happens. Assuming that event $E(x) = (E_1(x) \vee E_2(x))$ does not happen, both experiments will proceed identically the same and the output distributions of $Ev^{\widehat{\Psi}}(B, z)$ and $Ev^{\widetilde{\Psi}}(B, z)$ will be statistically close.

Claim 7.16. $\Pr_x[E_1(x)] \leq \epsilon/4$.

Proof. We bound this event in Hybrid 1. Recall that E_1 is the event that during the final execution of the obfuscation $\widehat{B}$ on a random input, a successful decryption or reveal query is asked for the first time for a ciphertext that was generated during the obfuscation phase but was not learned during the learning phase. For all $i \in [t]$, let $Q'_{B_i} = (Q_{B_i} \cup Q^h_{B_i}) \cap Q_O$ be the set of query-answer pairs generated by the i'th evaluation $Ev^{\Psi}(B, z_i)$ during the learning phase (Step 2) and are also generated during the obfuscation emulation phase (Step 1). Note that, since $t > \ell_O$ and $Q'_{B_i} \subseteq Q'_{B_{i+1}}$, the number of learning iterations that increase the size of the set of learned obfuscation encryption queries is at most ℓ_O. We say $t \xleftarrow{\$} [m]$ is bad if it is the case that $Q'_{B_t} \neq Q'_{B_{t+1}}$ (i.e. t is an index of a learning iteration that increases the size of the learned obfuscation encryption queries). This would imply that after t learning iterations in Hybrid 1, the real execution $Q'_{\widehat{B}} := Q'_{B_{t+1}}$ would contain a new unlearned query that was in Q_O. Thus, given that $m = 4\ell_O/\epsilon$, the probability (over the selection of t) that t is bad is at most $\ell_O/m < \epsilon/4$. □

Claim 7.17. $\Pr_x[E_2(x) \mid \overline{E}_1(x)] \leq \epsilon/4 + \mathrm{negl}(\kappa)$.

Proof. Recall that E_2 is the event that during the final execution of the obfuscation $\widehat{B}$ on a random input, a decryption or reveal query is asked for a ciphertext that was generated during the learning phase but was part of the hidden set and therefore never publicized. We will in fact first bound this event (assuming E_1 does not happen) in Hybrid 2.

For all $i \in [t]$, let $Q_{B_i} = (Q_{B_i} \cup Q^h_{B_i})$ be the set of (public and hidden) query-answer pairs generated by the i'th evaluation $Ev^{\Psi}(B, z_i)$ during the learning phase (Step 2) and $Q_{\widehat{B}}$ be the query-answer pairs of the final evaluation of the obfuscation. Note that in Hybrid 2, the real oracle Ψ is used and therefore each execution of $Ev^{\Psi}(B, z_i)$ is independent of the other executions. In particular, we can think of the final evaluation as iteration $t + 1$ of the learning phase with query-answer pairs $Q_{B_{t+1}} = Q_{\widehat{B}}$. Now, we can think of an intermediate hybrid with distribution $H^{2'}$ where the executions are randomly permuted in a way such that execution $t + 1$ in Hybrid 2 is now the first execution in Hybrid $2'$. Note that, since the executions are independent, the distributions H^2 and $H^{2'}$ are equivalent and it remains to show that E_2 is unlikely to happen in Hybrid $2'$. Specifically, we need to show that the first execution of the learning phase (with query-answer pairs $Q_{B_{t+1}}$) does not decrypt a ciphertext generated by any previous hidden queries in the learning phase. However, since this is the first execution, there exists no hidden queries from previous executions and it suffices to argue that it does not decrypt a ciphertext generated by any hidden queries $Q^h_{B_{t+1}}$ from its own execution.

Since E_1 does not happen and since the probability that $Ev^{\Psi}(B, z_{t+1})$ decrypts a ciphertext that was never encrypted during the whole process is negligible, the only ciphertexts that this execution will attempt to decrypt are ones that were generated during this very same execution. Therefore in Hybrid 2, E_2 only happens with negligible probability when a ciphertext that was never encrypted before is being decrypted. Now using Claim 7.18 we can conclude that this event in Hybrid 1 happens with probability at most $\epsilon/4 + \mathrm{negl}(\kappa)$.

□

Thus, using Claims 7.16 and 7.17, we find that $\Delta(H^0, H^1) \leq \epsilon/2 + \mathrm{negl}(\kappa)$.

□

Claim 7.18. $\Delta(H^1, H^2) \leq \dfrac{\epsilon}{4} + \mathrm{negl}(\kappa)$.

Proof. The only difference between the two hybrids is that in Hybrid 1, we do not forward to Ψ any queries of the form $\text{Eval}(c_1, ..., c_k)$ for which we do not know all of c_i, and instead we will use some simulated (fake) ciphertexts. Whereas in Hybrid 2, we do indeed execute them using the real oracle. Thus, while there may be hidden queries as a result of running F in Hybrid 2, the counterparts in Hybrid 1 do not have such hidden queries and are instead replaced with fake ciphertexts. As a result, we have to consider the cases where Rev, Dec or Eval might use any hidden ciphertexts in Hybrid 2 and/or fake ciphertexts in Hybrid 1.

Assume that the query-answer pairs so far during the execution of $\widehat{B}$ are the same in both hybrids. That is, we start with some $Q^1_{\widehat{B}} = Q^2_{\widehat{B}} = Q_{\widehat{B}}$ for the sake of proving inductively that any subsequent query-answer pairs are closely distributed between the two hybrids. Let q be a new (possibly indirect) query that is being asked by $Ev(B, z)$.

1. Determined queries: If q is a query of type $\text{Enc}(x)$ then it will answered the same in both hybrid using R. If q is of type $\text{Rev}(c)$ for any c then the answer will be the same in both hybrids. If q is of type $\text{Dec}(w, c)$ for any non-fake ciphertext c where $(x \mapsto c)_{\text{Enc}} \in Q_S \setminus Q^h_{\widehat{B}}$, then the answer will be the same in both hybrids. If q is of type $\text{Eval}(c_1, ..., c_k)$ whose answer can be determined by $Q_B \cup Q_{\widehat{B}}$ then the answer will be the same in both hybrids.

2. **Bad event 1.** If q is of type $\text{Eval}(c_1, ..., c_k)$ such that there exists $(x_i \mapsto c_i)_{\text{Enc}} \in Q_S \setminus (Q_B \cup Q_{\widehat{B}})$ for some $i \in [k]$ then this means that we are evaluating on some ciphertext that was not learned. In this case in Hybrid 1 the answer would be a set of fake ciphertexts generated by encrypting random messages under their respective attributes. However, in Hybrid 2 the answer would be a set of ciphertexts generated by encrypting shares of the true evaluation. Note that by the collision resistance property of H and the injectivity of Enc, the two sets of ciphertexts (the fake and non-fake) will be statistically close.

3. Suppose that q is of type $\text{Dec}(w, c)$. We list the different cases that can cause problems:

   (a) **Bad event 2.** There exists a query-answer pair $(x \mapsto c)_{\text{Enc}} \in Q^h_{\widehat{B}} \setminus (Q_B \cup Q_{\widehat{B}})$ that was generated as one of the hidden queries issued by some evaluation query during execution $\widehat{Ev}(\widehat{B}, z)$. In that case, in Hybrid 1, the answer would be $\perp$ with overwhelming probability since the encryption was never created there, whereas in Hybrid 2, the answer would be x. However we note that the probability of that happening in Hybrid 1 is negligible since it is equal to the event that a ciphertext was hit that has not been generated before.

   (b) **Bad event 3.** The query-answer pair $((c_1, ..., c_k) \mapsto (c'_1, ..., c'_p))_{\text{Eval}}$ is in $Q_{\widehat{B}}$, $c = c'_i$ for some $i \in [p]$, $\exists((a_j, m_j) \mapsto c_j)_{\text{Enc}} \notin Q_B \cup Q_{\widehat{B}}$ and $\mathsf{V}^{\Psi}(w, a'_i) = 1$ where $a'_i = \text{Rev}(c'_i)$. That is, we are decrypting one of the (fake) ciphertexts from the output of an evaluation query for which the underlying plaintexts of at least one of the ciphertext inputs to the evaluation query is unknown. Note that each fake ciphertext c'_i corresponds to an encryption query $(x'_i \mapsto c'_i)_{\text{Enc}} \in Q_{\widehat{B}}$ for some $x'_i = (a'_i, r_i)$ and independently chosen random string r_i. In that case, in Hybrid 1, the answer would be x'_i whereas in Hybrid 2 it would use the answer supplied by Ψ and output the answer of the evaluation $x_i = (a'_i, m'_i)$ where m'_i is an evaluated answer based on the messages encrypted by $(c_1, ..., c_k)$.

   For the case of FHE (or multi-key FHE), decrypting any fake ciphertext implies decrypting for the first time all of the input ciphertexts, which must have originated from Q_O

or Q_B^h since we need to know witnesses for all the input ciphertexts $(c_1, ..., c_k)$ in order to decrypt c. Thus, this event is similar to Case 8d of Claim 7.15 and so we refer to the proof there for this case.

For the more general case of spooky encryption, we may decrypt a fake ciphertext without needing to know witnesses for all the input ciphertexts and so we cannot argue that we are decrypting for the first time some input ciphertext. Nevertheless, all we obtain by decrypting a single (or even $p-1$) ciphertexts are individual shares of the evaluation result, which are identically distributed to random values r_i as long as we do not have witnesses for *all* the input ciphertexts (if we do, we are back to the multi-key FHE case).

Using the same reasoning as Claims 7.16 and 7.17, we find that $\Delta(H^1, H^2) \leq \epsilon/4 + \text{negl}(\kappa)$.

□

Using Claims 7.15 and 7.18 we can finally conclude the proof of approximate correctness since we can now show that $\Delta(H^0, H^1) + \Delta(H^1, H^2) \leq 3\epsilon/4 + \text{negl}(\kappa) \leq \epsilon$.

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 Ψ) that works as follows: given an obfuscated circuit B in the Ψ ideal model, it runs the learning procedure as shown in Step 2 of the new obfuscator $\widehat{iO}$ to learn the heavy queries Q_B then outputs $\widehat{B} = (B, Q_B)$. Note that this distribution is statistically close to the output of the real execution of $\widehat{iO}$ and, therefore, security follows.

□

8 Primitives Implied by Our Variants of Witness Encryption

In this section, we will show that extended witness encryption (or one of its variants) implies extended versions of several encryption primitives of the all-or-nothing flavor. Recall that we previously argued that witness encryption and its considered variants do not imply indistinguishability obfuscation in a monolithic way. This allows us to conclude a monolithic separation between the studied all-or-nothing encryption primitives and indistinguishability obfuscation.

8.1 Predicate Encryption

In this section, we will show a monolithic construction of extended PE from EIHWE and one-way function. Since, one-way functions imply digital signatures in a black-box manner, we will assume that there exists a signature scheme $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ where these procedures only make black-box calls to the OWF. Let $(\text{WEnc}, \text{WDec}_{\mathsf{V}})$ be an EIHWE scheme where V is the universal circuit evaluator (see Definition **??**) that is allowed to have OWF gates, and WEnc and WDec_{V} gates. The relation V is defined such that $\mathsf{V}((k, sk_k), C_{\mathsf{MPK},a,m}) = C_{\mathsf{MPK},a,m}(k, sk_k)$ where circuit $C_{\mathsf{MPK},a,m}$ is defined as follows:

$$C_{\mathsf{MPK},a,m}(k, sk_k) := (\mathsf{P}(k, a) = 1 \wedge \mathsf{Verify}(\mathsf{MPK}, k, sk_k) = 1)\,.$$

Then, our extended PE scheme for the universal class of predicates $P_{K,A}$ (computed by Turing Machine P as in Definition 3.6) and message space M corresponds to the PPT algorithms

$(\mathsf{Setup}, \mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec}_{\mathsf{P}})$ defined below. Note that we are constructing an extended PE scheme and therefore P is allowed to have OWF, $\mathsf{Setup}, \mathsf{KGen}, \mathsf{Enc}$, and $\mathsf{Dec}_{\mathsf{P}}$ gates.

- $\mathsf{Setup}(1^\kappa)$: outputs $(\mathsf{MPK}, \mathsf{MSK})$ where $(\mathsf{MPK}, \mathsf{MSK}) \leftarrow \mathsf{Gen}(1^\kappa)$.
- $\mathsf{KGen}(\mathsf{MSK}, k)$: given $k \in K$ and the master secret key $\mathsf{MSK} \in \{0,1\}^n$, outputs the decryption key $sk_k = \mathsf{Sign}(\mathsf{MSK}, k)$. If $k = \epsilon$, it outputs ϵ.
- $\mathsf{Enc}(\mathsf{MPK}, (m, a))$: outputs $c = \mathrm{WEnc}(C_{\mathsf{MPK},a,m}, (a, m))$ where $C_{\mathsf{MPK},a,m}$ is defined above.
- $\mathsf{Dec}_{\mathsf{P}}(sk_k, c)$: given a secret key sk_k for $k \in K$ and a ciphertext c, obtain $(C_{\mathsf{MPK},a,m}, a, m) = \mathrm{WDec}_{\mathsf{V}}((k, sk_k), c)$, and output m.

Specifying V given P. Since we are constructing extended predicate encryption, our P is allowed to have gates of OWFs, $\mathsf{Setup}, \mathsf{KGen}, \mathsf{Enc}$, and $\mathsf{Dec}_{\mathsf{P}}$ planted in it. Next, observe from the scheme that V contains one P gate and one Verify gate. Also note that P does not make any calls to V.

Next we argue that both the P gate and the one Verify gate can be simplified to OWF, WEnc, and $\mathrm{WDec}_{\mathsf{V}}$ gates that V is allowed to have. We modify V as follows.

1. We embed P in V. Note that V now has all the gates P had and a Verify gate. Specifically, V has OWF, $\mathsf{Verify}, \mathsf{Setup}, \mathsf{KGen}, \mathsf{Enc}$, and $\mathsf{Dec}_{\mathsf{P}}$ gates.
2. Next, we syntactically replace Setup gates with Gen gates, KGen gates with Sign gates, Enc gates with WEnc gates and $\mathsf{Dec}_{\mathsf{P}}$ gates with $\mathrm{WDec}_{\mathsf{V}}$ gates. Note that this replacement will require some additional code changes in V which all depend on the above described construction.
3. Next, since we use a known construction of the signature scheme $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ and it is black-box in the use of OWFs, we can replace these gates by just OWF gates along with some additional code that depends on the used signature scheme. Observe that we have reduced all gates in V to just OWF, WDec, and $\mathrm{WDec}_{\mathsf{V}}$ gates.

Lemma 8.1. *Extended fully-secure PE (Definition 3.6) is implied by extended EIHWE and OWFs.*

Proof. Correctness of the above scheme follows from the observation that a witness encryption ciphertext along with a signature on k, namely sk_k, always yields the encrypted message whenever $\mathsf{P}(k, a) = 1$. Next we prove security.

Let A be a computationally bounded adversary that asks a polynomial number of secret-key queries and breaks the security of the PE scheme. In other words, for some polynomial $p(.)$,

$$\Pr[\mathrm{IND}_A^{\mathrm{PE}}(1^\kappa) = 1] \geq \frac{1}{2} + \frac{1}{p(\kappa)}$$

where $\mathrm{IND}_A^{\mathrm{PE}}$ is the following experiment (recalled from Figure 2):

Experiment $\mathrm{IND}_A^{\mathrm{PE}}(1^\kappa)$:
1. $(\mathsf{MSK}, \mathsf{MPK}) \leftarrow \mathrm{Setup}(1^\kappa))$
2. $(x_0, x_1) \leftarrow A^{\mathrm{KGen}(\mathsf{MSK},.)}(\mathsf{MPK})$ where $x_b = (a_b, m_b)$ for $b \in \{0,1\}$, $|a_0| = |a_1|$ and $|m_0| = |m_1|$

and for each prior query k we require that $\mathsf{P}(k, a_0) = \mathsf{P}(k, a_1) = 0$
3. $b \xleftarrow{\$} \{0,1\}$
4. $c \leftarrow \mathrm{Enc}(\mathsf{MPK}, x_b)$
5. $b' \leftarrow A^{\mathrm{KGen}(\mathsf{MSK},.)}(\mathsf{MPK}, c)$ where for each query k we require that $\mathsf{P}(k, a_0) = \mathsf{P}(k, a_1) = 0$
6. Output 1 if $b = b'$ and 0 otherwise.

At a high level, we would like to use the attacker A above to contradict the security of the signature scheme. Towards this, our approach will be to use the extractor E of ex-EIHWE that translates A's ability to distinguish ciphertexts to extracting a witness, which in our case is a forgery. However, a technical issue is that E might rewind A and A might fail on those correlated executions. We solve this issue by a standard probability argument as explained next.

Let $\mathcal{T}$ be the transcript of the inputs to A in steps 1 and 2 above — namely, $\mathcal{T} = (\mathsf{MPK}, \{sk_k\}, x_0, x_1)$ where $\{sk_k\}$ is the collection of secret-keys that A obtains in step 2 above. Then, for the machine $A(\mathcal{T})$ we define the following experiment:

Experiment $\mathrm{IND}^{\text{PE-partial}}_{A(\mathcal{T})}(1^\kappa, \mathsf{MPK}, \mathsf{MSK})$:
1. $b \xleftarrow{\$} \{0,1\}$
2. $c \leftarrow \mathrm{Enc}(\mathsf{MPK}, x_b)$
3. $b' \leftarrow A(\mathcal{T})^{\mathrm{KGen}(\mathsf{MSK},.)}(\mathsf{MPK}, c)$ where for each query k we require that $\mathsf{P}(k, a_0) = \mathsf{P}(k, a_1) = 0$
4. Output 1 if $b = b'$ and 0 otherwise.

By an averaging argument we have that if $\Pr[\mathrm{IND}^{\mathrm{PE}}_A(1^\kappa) = 1] \geq \frac{1}{2} + \frac{1}{p(\kappa)}$ then for a non-negligible fraction of choices of $(\mathcal{T}, \mathsf{MPK}, \mathsf{MSK})$ we have that $\Pr[\mathrm{IND}^{\text{PE-partial}}_{A(\mathcal{T})}(1^\kappa, \mathsf{MPK}, \mathsf{MSK}) = 1] \geq \frac{1}{2} + \frac{1}{2p(\kappa)}$.

Towards contradiction, we will now show that given A we can break the existential unforgeability of the signature scheme under chosen message attack. On receiving the public-key MPK from the challenger we prepare the machine $A(\mathcal{T})$ (by including signatures on keys of A's choice in $\mathcal{T}$). Next by Definition 6.2 we have that there exists an a black-box extractor $E^{A(\mathcal{T})}(a_0, a_1)$ which with non-negligible probability outputs a signature on k^* for some choice of k^* such that $\mathsf{P}(k^*, a_0) = 1$ or $\mathsf{P}(k^*, a_1) = 1$. This breaks the security of the signature scheme.

Note that during the execution of $E^{A(\mathcal{T})}(a_0, a_1)$, A could ask for more signature queries for any k such that $\mathsf{P}(k, a_0) = \mathsf{P}(k, a_1) = 0$. However, these queries can be answered with the help of the challenger.

□

8.2 Spooky Encryption

In this section, we will show a monolithic construction of extended spooky encryption from IRHWE and PRG. Let $(\mathrm{WEnc}, \mathrm{WDec}_{\mathsf{V}}, \mathrm{WEval}_{\mathsf{F}})$ be an IRHWE scheme where F is a universal circuit evaluator and G be a length doubling pseudorandom generator. The relation V is defined such that

$\mathsf{V}(w, C_{\mathsf{MPK}}) = C_{\mathsf{MPK}}(w)$ where circuit C_{MPK} is defined as follows:

$$C_{\mathsf{MPK}}(w) := (\mathsf{MPK} = G(w)) .$$

Then our extended spooky encryption scheme $(\mathrm{Setup}, \mathrm{Enc}, \mathrm{Eval}_{\mathsf{F}_s}, \mathrm{Dec})$ (where F_s denotes the universal circuit evaluation function for the spooky encryption scheme) is defined as follows:

- $\mathrm{Setup}(1^\kappa)$: outputs $(\mathsf{MPK} = G(s), \mathsf{MSK} = s)$ where s is a randomly sampled seed of the pseudorandom generator.
- $\mathrm{Enc}(\mathsf{MPK}, m)$: outputs ciphertext $c = \mathrm{WEnc}(C_{\mathsf{MPK}}, m)$ where C_{MPK} is defined above.
- $\mathrm{Eval}_{\mathsf{F}_s}(f, (\mathsf{MPK}_1, c_1), \ldots, (\mathsf{MPK}_t, c_t))$: output $\mathrm{WEval}_{\mathsf{F}}(f, c_1, \ldots c_t)$.
- $\mathrm{Dec}(\mathsf{MSK}, c)$: given a secret key MSK and a ciphertext c, output $\mathrm{WDec}_{\mathsf{V}}(\mathsf{MSK}, c)$.

Specifying V. Observe that V in the extended spooky encryption scheme above only has one PRG gate that the extended IRHWE scheme supports.

Specifying F_s. Since we are constructing extended spooky encryption, F_s is allowed to have gates of $\mathrm{Setup}, \mathrm{Enc}, \mathrm{Dec}$, and $\mathrm{Eval}_{\mathsf{F}_s}$ planted in it. We start by observing that for any F_s we can syntactically replace each Setup gate with a PRG gate, each Enc gate with a WEnc gate, each $\mathrm{Eval}_{\mathsf{F}_s}$ gate with a $\mathrm{WEval}_{\mathsf{F}}$ gate, and each Dec gate with a $\mathrm{WDec}_{\mathsf{V}}$ gate. This change requires some additional code depending on the above described construction. This leaves us with only PRG, WEnc, $\mathrm{WDec}_{\mathsf{V}}$, and $\mathrm{WEval}_{\mathsf{F}}$ gates that F of extended IRWHE supports. Thus the F_s supports gates as required by extended spooky encryption scheme.

Lemma 8.2. *Extended spooky encryption (Definition 3.10) is implied by ex-IRHWE and PRG.*

Proof. Correctness of the above scheme follows directly from the correctness of the IRHWE scheme. For security, we need to show that for any PPT adversary A, there exists a negligible function $\mathrm{negl}(.)$ such that the following holds:

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

where $(\mathsf{MSK}, \mathsf{MPK}) \leftarrow \mathrm{Setup}(1^\kappa)$ and the probability is over the randomness of A and Enc. This follows by a simple argument as follows.

- H_0: This hybrid corresponds to the distribution $(\mathsf{MPK}, \mathrm{Enc}(\mathsf{MPK}, 0))$ where $(\mathsf{MSK}, \mathsf{MPK}) \leftarrow \mathrm{Setup}(1^\kappa)$.
- H_1: In this hybrid, we change how MPK is generated. Instead of generating MPK as the output of G, we sample MPK as a uniformly random string.

  Indistinguishability between H_0 and H_1 follows from the pseudorandomness of the PRG G.
- H_2: In this hybrid, instead of encrypting 0 we encrypt 1. Namely, the ciphertext is generated as $\mathrm{Enc}(\mathsf{MPK}, 1)$ instead of $\mathrm{Enc}(\mathsf{MPK}, 0)$.

  Since MPK is a chosen uniformly at random therefore except with negligible probability we have that $\forall w, \mathsf{MPK} \neq G(w)$. Hence, by the security of witness encryption we have that the distribution $\mathrm{Enc}(\mathsf{MPK}, 1)$ is computationally indistinguishable from $\mathrm{Enc}(\mathsf{MPK}, 0)$.

- H_3: In this hybrid, we switch back to generating MPK honestly (i.e., as the output of the pseudorandom generator).

  Indistinguishability between H_2 and H_3 follows from the pseudorandomness of the PRG G.

Observe that hybrid H_3 corresponds to the distribution $(\mathsf{MPK}, \mathrm{Enc}(\mathsf{MPK}, 1))$ where $(\mathsf{MSK}, \mathsf{MPK}) \leftarrow \mathrm{Setup}(1^\kappa)$. This concludes the proof. □

8.3 Attribute-Based FHE

In this section, we will show a construction of extended Attribute Based FHE (ABFHE) from extended extractable instance-revealing homomorphic witness encryption (EIRHWE) and one-way functions (OWFs). Since one-way functions imply digital signatures in a black-box manner, we will assume that there exists a signature scheme $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ where these procedures only make black-box calls to the OWF. Let $(\mathrm{WEnc}, \mathrm{WRev}, \mathrm{WDec}_{\mathsf{V}}, \mathrm{WEval}_{\mathsf{F}})$ be the ex-EIRHWE primitive and V and F are universal circuit evaluators that are allowed to have OWF gates, and $\mathrm{WEnc}, \mathrm{WDec}_{\mathsf{V}}$, and $\mathrm{WEval}_{\mathsf{F}}$ oracle gates. The relation V is defined such that $\mathsf{V}((k, sk_k), C_{\mathsf{MPK},a}) = C_{\mathsf{MPK},a}(k, sk_k)$ where circuit $C_{\mathsf{MPK},a}$ is defined as follows:

$$C_{\mathsf{MPK},a}(k, sk_k) := (\mathsf{P}(k, a) = 1 \wedge \mathsf{Verify}(\mathsf{MPK}, k, sk_k) = 1)\,.$$

Then, our extended ABFHE scheme for the universal predicate class $P_{K,A}$ (computed by machine P) and message space M corresponds to the PPT algorithms $(\mathrm{Setup}, \mathrm{KGen}, \mathrm{Enc}, \mathrm{Dec}_{\mathsf{P}}, \mathrm{Eval}_{\mathsf{F}_s})$ defined below. Note that we are constructing an extended ABFHE, therefore P and F_s are allowed to have OWF, $\mathrm{Setup}, \mathrm{KGen}, \mathrm{Enc}, \mathrm{Dec}_{\mathsf{P}}$ and $\mathrm{Eval}_{\mathsf{F}_s}$ gates.

- $\mathrm{Setup}(1^\kappa)$: outputs $(\mathsf{MPK}, \mathsf{MSK})$ where $(\mathsf{MPK}, \mathsf{MSK}) \leftarrow \mathsf{Gen}(1^\kappa)$.
- $\mathrm{KGen}(\mathsf{MSK}, k)$: given $k \in K$ and the master secret key $\mathsf{MSK} \in \{0,1\}^n$, outputs the decryption key $sk_k = \mathsf{Sign}(\mathsf{MSK}, k)$. If $k = \epsilon$, it outputs ϵ.
- $\mathrm{Enc}(\mathsf{MPK}, (m, a))$: outputs ciphertext $c = \mathrm{WEnc}(C_{\mathsf{MPK},a}, m)$ where $C_{\mathsf{MPK},a}$ is defined above.
- $\mathrm{Eval}_{\mathsf{F}_s}(f, \mathsf{MPK}, c_1, \ldots, c_t)$: If $\mathrm{WRev}(c_1) = \mathrm{WRev}(c_2) = \cdots = \mathrm{WRev}(c_t)$[19] then output $\mathrm{WEval}_{\mathsf{F}}(f, c_1, \ldots, c_t)$ and otherwise output $\perp$.
- $\mathrm{Dec}_{\mathsf{P}}(sk_k, c)$: given a secret key sk_k for $k \in K$ and a ciphertext c, output $\mathrm{WDec}_{\mathsf{V}}((k, sk_k), c)$.

Specifying V given P. Since we are constructing extended ABFHE, our P is allowed to have gates of $\mathrm{OWF}, \mathrm{Setup}, \mathrm{KGen}, \mathrm{Enc}, \mathrm{Eval}_{\mathsf{F}_s}$ and $\mathrm{Dec}_{\mathsf{P}}$ planted in it. Observe from the scheme that V contains one P gate and one Verify gate. Next we argue that both these gates can be simplified to $\mathrm{OWF}, \mathrm{WEnc}, \mathrm{WDec}_{\mathsf{V}}$, and $\mathrm{WEval}_{\mathsf{F}}$ gates that ex-EIRHWE supports. We modify V as follows.

1. We embed P in V. Note that V now has all the gates P had and a Verify gate. Specifically, V has $\mathrm{OWF}, \mathsf{Verify}, \mathrm{Setup}, \mathrm{KGen}, \mathrm{Enc}, \mathrm{Eval}_{\mathsf{F}_s}$ and $\mathrm{Dec}_{\mathsf{P}}$ gates.

[19] Here, WRev is the reveal function of the IRWHE that we ignored in the rest of the exposition. This function is only needed at this point.

2. Next, we syntactically replace Setup gates with Gen gates, KGen gates with Sign gates, Enc gates with $\mathrm{WDec}_{\mathsf{V}}$ gates, $\mathrm{Eval}_{\mathsf{F}_s}$ gates with $\mathrm{WEval}_{\mathsf{F}}$ gates and $\mathrm{Dec}_{\mathsf{P}}$ gates with $\mathrm{WDec}_{\mathsf{V}}$ gates. Note that this replacement will require some additional code changes in V which all depend on the above described construction.

3. Next, since we use a known construction of the signature scheme $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ and it is black-box in the use of OWFs, we can replace these gates by just OWF gates along with some additional code that depends on the used signature scheme. Observe that we have reduced all gates in V to just OWF, WEnc, $\mathrm{WEval}_{\mathsf{F}}$ and $\mathrm{WDec}_{\mathsf{V}}$ gates.

Specifying F given F_s. Since we are constructing extended ABFHE, $\mathrm{Eval}_{\mathsf{F}_s}$ is allowed to have gates of OWFs, Setup, KGen, Enc, $\mathrm{Dec}_{\mathsf{P}}$, and $\mathrm{Eval}_{\mathsf{F}_s}$ planted in it. We start by observing that for any F_s we can syntactically replace each Setup and KGen gate with a OWF gates, each Enc, $\mathrm{Eval}_{\mathsf{F}_s}$ and $\mathrm{Dec}_{\mathsf{P}}$ with WEnc, $\mathrm{WEval}_{\mathsf{F}}$ and $\mathrm{WDec}_{\mathsf{V}}$, respectively.In making this change we need to add some additional code to F depending on the above described construction and the code of the signature scheme. This leaves us with only OWF, WEnc, $\mathrm{WDec}_{\mathsf{V}}$, and $\mathrm{WEval}_{\mathsf{F}}$ gates that F of ex-EIRHWE supports. Thus the above described construction supports gates as required by the extended ABFHE scheme.

Lemma 8.3. *Extended fully-secure ABFHE scheme is implied by ex-EIRHWE and OWFs.*

Proof. Note that security game of the ABFHE scheme does not involve the $\mathrm{Eval}_{\mathsf{F}_s}$ function, which only affects functionality. Therefore, the proof of security of this claim is essentially the same as the proof of Lemma 8.1. The only difference is that we are now using ex-EIRHWE instead of ex-EIHWE. □

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

References

[AB15] Benny Applebaum and Zvika Brakerski. Obfuscating circuits via composite-order graded encoding. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015: 12th Theory of Cryptography Conference, Part II*, volume 9015 of *Lecture Notes in Computer Science*, pages 528–556, Warsaw, Poland, March 23–25, 2015. Springer, Heidelberg, Germany. 3

[ABG+13] Prabhanjan Ananth, Dan Boneh, Sanjam Garg, Amit Sahai, and Mark Zhandry. Differing-inputs obfuscation and applications. Cryptology ePrint Archive, Report 2013/689, 2013. http://eprint.iacr.org/2013/689. 15

[AGIS14] Prabhanjan Vijendra Ananth, Divya Gupta, Yuval Ishai, and Amit Sahai. Optimizing obfuscation: Avoiding Barrington's theorem. In Gail-Joon Ahn, Moti Yung, and Ninghui Li, editors, *ACM CCS 14: 21st Conference on Computer and Communications Security*, pages 646–658, Scottsdale, AZ, USA, November 3–7, 2014. ACM Press. 3

[AJ15] Prabhanjan Ananth and Abhishek Jain. Indistinguishability obfuscation from compact functional encryption. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *Advances in Cryptology – CRYPTO 2015, Part I*, volume 9215 of *Lecture Notes in Computer Science*, pages 308–326, Santa Barbara, CA, USA, August 16–20, 2015. Springer, Heidelberg, Germany. 3, 4, 5, 27, 31

[AJS15] Prabhanjan Ananth, Abhishek Jain, and Amit Sahai. Indistinguishability obfuscation from functional encryption for simple functions. Cryptology ePrint Archive, Report 2015/730, 2015. http://eprint.iacr.org/2015/730. 4, 5, 27

[App14] Benny Applebaum. Bootstrapping obfuscators via fast pseudorandom functions. In Palash Sarkar and Tetsu Iwata, editors, *Advances in Cryptology – ASIACRYPT 2014, Part II*, volume 8874 of *Lecture Notes in Computer Science*, pages 162–172, Kaoshiung, Taiwan, R.O.C., December 7–11, 2014. Springer, Heidelberg, Germany. 5

[AS15] 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. 4, 7, 23, 27

[AS16] 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. 4, 23, 27

[BBF13] 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. 7, 22

[BBF16] 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/. 7, 11, 23, 25, 26

[BCP14] Elette Boyle, Kai-Min Chung, and Rafael Pass. On extractability obfuscation. In Yehuda Lindell, editor, *TCC 2014: 11th Theory of Cryptography Conference*, volume 8349 of *Lecture Notes in Computer Science*, pages 52–73, San Diego, CA, USA, February 24–26, 2014. Springer, Heidelberg, Germany. 15

[BGI+01] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In Joe Kilian, editor, *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 1–18, Santa Barbara, CA, USA, August 19–23, 2001. Springer, Heidelberg, Germany. 3

[BGK+14] Boaz Barak, Sanjam Garg, Yael Tauman Kalai, Omer Paneth, and Amit Sahai. Protecting obfuscation against algebraic attacks. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology – EUROCRYPT 2014*, volume 8441 of *Lecture Notes in Computer Science*, pages 221–238, Copenhagen, Denmark, May 11–15, 2014. Springer, Heidelberg, Germany. 3

[BKSY11] 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. 4, 27

[BMSZ15] Saikrishna Badrinarayanan, Eric Miles, Amit Sahai, and Mark Zhandry. Post-zeroizing obfuscation: The case of evasive circuits. Cryptology ePrint Archive, Report 2015/167, 2015. http://eprint.iacr.org/2015/167. 3

[Bor09] Émile Borel. Les probabilités dénombrables et leurs applications arithmétiques. *Rendiconti del Circolo Matematico di Palermo (1884-1940)*, 27(1):247–271, 1909. 23

[BP16] 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. 3

[BR14] Zvika Brakerski and Guy N. Rothblum. Virtual black-box obfuscation for all circuits via generic graded encoding. In Yehuda Lindell, editor, *TCC 2014: 11th Theory of Cryptography Conference*, volume 8349 of *Lecture Notes in Computer Science*, pages 1–25, San Diego, CA, USA, February 24–26, 2014. Springer, Heidelberg, Germany. 3

[BSW07] John Bethencourt, Amit Sahai, and Brent Waters. Ciphertext-policy attribute-based encryption. In *2007 IEEE Symposium on Security and Privacy*, pages 321–334, Oakland, CA, USA, May 20–23, 2007. IEEE Computer Society Press. 22, 28

[BSW10] Dan Boneh, Amit Sahai, and Brent Waters. Functional Encryption: Definitions and Challenges. Technical Report 2005/328, Cryptology ePrint Archive, 2010. http://eprint.iacr.org/2010/543. 18, 19

[BV11a] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In Rafail Ostrovsky, editor, *52nd Annual Symposium on Foundations of Computer Science*, pages 97–106, Palm Springs, CA, USA, October 22–25, 2011. IEEE Computer Society Press. 3

[BV11b] Zvika Brakerski and Vinod Vaikuntanathan. Fully homomorphic encryption from ring-LWE and security for key dependent messages. In Phillip Rogaway, editor, *Advances in Cryptology – CRYPTO 2011*, volume 6841 of *Lecture Notes in Computer Science*, pages 505–524, Santa Barbara, CA, USA, August 14–18, 2011. Springer, Heidelberg, Germany. 3

[BV15] Nir Bitansky and Vinod Vaikuntanathan. Indistinguishability obfuscation from functional encryption. In Venkatesan Guruswami, editor, *56th Annual Symposium on Foundations of Computer Science*, pages 171–190, Berkeley, CA, USA, October 17–20, 2015. IEEE Computer Society Press. 3, 4, 5, 27, 31

[BZ14] Dan Boneh and Mark Zhandry. Multiparty key exchange, efficient traitor tracing, and more from indistinguishability obfuscation. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology – CRYPTO 2014, Part I*, volume 8616 of *Lecture Notes in Computer Science*, pages 480–499, Santa Barbara, CA, USA, August 17–21, 2014. Springer, Heidelberg, Germany. 4

[Can17] Francesco Paolo Cantelli. Sulla probabilita come limite della frequenza. *Atti Accad. Naz. Lincei*, 26(1):39–45, 1917. 23

[CGH+15] 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, *Advances in Cryptology – CRYPTO 2015, Part I*, volume 9215 of *Lecture Notes in Computer Science*, pages 247–266, Santa Barbara, CA, USA, August 16–20, 2015. Springer, Heidelberg, Germany. 3

[CHL+15] 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, *Advances in Cryptology – EUROCRYPT 2015, Part I*, volume 9056 of *Lecture Notes in Computer Science*, pages 3–12, Sofia, Bulgaria, April 26–30, 2015. Springer, Heidelberg, Germany. 3

[CKP15] Ran Canetti, Yael Tauman Kalai, and Omer Paneth. On obfuscation with random oracles. Cryptology ePrint Archive, Report 2015/048, 2015. http://eprint.iacr.org/. 7, 11, 12, 13, 23, 25, 37, 38, 48, 62, 63

[CLLT15] 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. 3

[CLT13] Jean-Sébastien Coron, Tancrède Lepoint, and Mehdi Tibouchi. Practical multilinear maps over the integers. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013, Part I*, volume 8042 of *Lecture Notes in Computer Science*, pages 476–493, Santa Barbara, CA, USA, August 18–22, 2013. Springer, Heidelberg, Germany. 3

[CLTV15] Ran Canetti, Huijia Lin, Stefano Tessaro, and Vinod Vaikuntanathan. Obfuscation of probabilistic circuits and applications. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015: 12th Theory of Cryptography Conference, Part II*, volume 9015 of *Lecture Notes in Computer Science*, pages 468–497, Warsaw, Poland, March 23–25, 2015. Springer, Heidelberg, Germany. 5

[CM15] 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, *Advances in Cryptology – CRYPTO 2015, Part II*, volume 9216 of *Lecture Notes in Computer Science*, pages 630–656, Santa Barbara, CA, USA, August 16–20, 2015. Springer, Heidelberg, Germany. 3

[DHRW16a] Yevgeniy Dodis, Shai Halevi, Ron D. Rothblum, and Daniel Wichs. Spooky encryption and its applications. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology – CRYPTO 2016, Part III*, volume 9816 of *Lecture Notes in Computer Science*, pages 93–122, Santa Barbara, CA, USA, August 14–18, 2016. Springer, Heidelberg, Germany. 3

[DHRW16b] Yevgeniy Dodis, Shai Halevi, Ron D. Rothblum, and Daniel Wichs. *Spooky Encryption and Its Applications*, pages 93–122. Springer Berlin Heidelberg, Berlin, Heidelberg, 2016. 21

[DS16] 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

[DTT10] Anindya De, Luca Trevisan, and Madhur Tulsiani. *Time Space Tradeoffs for Attacks against One-Way Functions and PRGs*, pages 649–665. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010. 46

[FFS88] Uriel Feige, Amos Fiat, and Adi Shamir. Zero-knowledge proofs of identity. *Journal of Cryptology*, 1(2):77–94, 1988. 8

[FS86] A. Fiat and A. Shamir. How to Prove Yourself: Practical Solutions to Identification and Signature Problems. In *Crypto '86*, pages 186–194, 1986. LNCS No. 263. 8

[Gen09a] Craig Gentry. *A fully homomorphic encryption scheme*. PhD thesis, Stanford University, 2009. crypto.stanford.edu/craig. 5, 31

[Gen09b] Craig Gentry. Fully homomorphic encryption using ideal lattices. In Michael Mitzenmacher, editor, *41st Annual ACM Symposium on Theory of Computing*, pages 169–178, Bethesda, MD, USA, May 31 – June 2, 2009. ACM Press. 3, 4

[GGH13a] Sanjam Garg, Craig Gentry, and Shai Halevi. Candidate multilinear maps from ideal lattices. In Thomas Johansson and Phong Q. Nguyen, editors, *Advances in Cryptology – EUROCRYPT 2013*, volume 7881 of *Lecture Notes in Computer Science*, pages 1–17, Athens, Greece, May 26–30, 2013. Springer, Heidelberg, Germany. 3

[GGH+13b] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *54th Annual Symposium on Foundations of Computer Science*, pages 40–49, Berkeley, CA, USA, October 26–29, 2013. IEEE Computer Society Press. 3, 5

[GGH15] Craig Gentry, Sergey Gorbunov, and Shai Halevi. Graph-induced multilinear maps from lattices. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015: 12th Theory of Cryptography Conference, Part II*, volume 9015 of *Lecture Notes in Computer Science*, pages 498–527, Warsaw, Poland, March 23–25, 2015. Springer, Heidelberg, Germany. 3

[GGSW13] Sanjam Garg, Craig Gentry, Amit Sahai, and Brent Waters. Witness encryption and its applications. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th Annual ACM Symposium on Theory of Computing*, pages 467–476, Palo Alto, CA, USA, June 1–4, 2013. ACM Press. 6, 15, 16

[GL89] 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. 35, 59

[GMM+16] Sanjam Garg, Eric Miles, Pratyay Mukherjee, Amit Sahai, Akshayaram Srinivasan, and Mark Zhandry. Secure obfuscation in a weak multilinear map model. Cryptology ePrint Archive, Report 2016/817, 2016. http://eprint.iacr.org/2016/817. 3

[GPS16] Sanjam Garg, Omkant Pandey, and Akshayaram Srinivasan. Revisiting the cryptographic hardness of finding a nash equilibrium. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology – CRYPTO 2016, Part II*, volume 9815 of *Lecture Notes in Computer Science*, pages 579–604, Santa Barbara, CA, USA, August 14–18, 2016. Springer, Heidelberg, Germany. 4

[GPSW06] 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: 13th Conference on Computer and Communications Security*, pages 89–98, Alexandria, Virginia, USA, October 30 – November 3, 2006. ACM Press. Available as Cryptology ePrint Archive Report 2006/309. 22, 28

[GPSZ16] 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. 4

[GR07] Shafi Goldwasser and Guy N. Rothblum. *On Best-Possible Obfuscation*, pages 194–213. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007. 25

[GS01] Geoffrey Grimmett and David Stirzaker. *Probability and random processes*. Oxford university press, 2001. 23

[GSW13] 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, *Advances in Cryptology – CRYPTO 2013, Part I*, volume 8042 of *Lecture Notes in Computer Science*, pages 75–92, Santa Barbara, CA, USA, August 18–22, 2013. Springer, Heidelberg, Germany. 3

[GVW13] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Attribute-based encryption for circuits. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th Annual ACM Symposium on Theory of Computing*, pages 545–554, Palo Alto, CA, USA, June 1–4, 2013. ACM Press. 3

[GVW15] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Predicate encryption for circuits from LWE. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *Advances in Cryptology – CRYPTO 2015, Part II*, volume 9216 of *Lecture Notes in Computer Science*, pages 503–523, Santa Barbara, CA, USA, August 16–20, 2015. Springer, Heidelberg, Germany. 3

[GW11] 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

[HJ15] Yupu Hu and Huiwen Jia. Cryptanalysis of GGH map. Cryptology ePrint Archive, Report 2015/301, 2015. http://eprint.iacr.org/2015/301. 3

[Hol15] Thomas Holenstein. Complexity theory, 2015. http://www.complexity.ethz.ch/education/Lectures/ComplexityFS15/skript_printable.pdf. 24

[IR89] Russell Impagliazzo and Steven Rudich. Limits on the provable consequences of one-way permutations. In *21st Annual ACM Symposium on Theory of Computing*, pages 44–61, Seattle, WA, USA, May 15–17, 1989. ACM Press. 3, 4, 7, 8, 22

[LATV12] Adriana López-Alt, Eran Tromer, and Vinod Vaikuntanathan. On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In *Proceedings of the Forty-fourth Annual ACM Symposium on Theory of Computing*, STOC '12, pages 1219–1234, New York, NY, USA, 2012. ACM. 20, 21

[Lin16] Huijia Lin. Indistinguishability obfuscation from constant-degree graded encoding schemes. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology – EUROCRYPT 2016, Part I*, volume 9665 of *Lecture Notes in Computer Science*, pages 28–57, Vienna, Austria, May 8–12, 2016. Springer, Heidelberg, Germany. 3, 5

[LV16] 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. 3

[MMN15] 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/. 7, 11, 12, 13, 23

[MMN+16a] 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. 7, 11, 23, 24, 25

[MMN+16b] 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. 7, 11, 23, 24, 25

[MSW14] Eric Miles, Amit Sahai, and Mor Weiss. Protecting obfuscation against arithmetic attacks. Cryptology ePrint Archive, Report 2014/878, 2014. http://eprint.iacr.org/2014/878. 3

[MSZ16] 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. 3

[MW16] Pratyay Mukherjee and Daniel Wichs. Two round multiparty computation via multi-key FHE. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology – EUROCRYPT 2016, Part II*, volume 9666 of *Lecture Notes in Computer Science*, pages 735–763, Vienna, Austria, May 8–12, 2016. Springer, Heidelberg, Germany. 3, 20, 21

[Nao03] Moni Naor. On cryptographic assumptions and challenges. *Lecture Notes in Computer Science*, 2729:96–109, 2003. 7

[Pas11] 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

[Pas15] 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/. 7, 11, 12, 23

[PS16] Chris Peikert and Sina Shiehian. Multi-key FHE from LWE, revisited. Cryptology ePrint Archive, Report 2016/196, 2016. http://eprint.iacr.org/2016/196. 3

[PTV11] Rafael Pass, Wei-Lung Dustin Tseng, and Muthuramakrishan Venkitasubramaniam. Towards non-black-box separations in cryptography. TCC, 2011. 7

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *37th Annual ACM Symposium on Theory of Computing*, pages 84–93, Baltimore, MA, USA, May 22–24, 2005. ACM Press. 3

[RTV04] 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. 3, 4, 7, 8, 16, 22, 27

[SW05] Amit Sahai and Brent R. Waters. Fuzzy identity-based encryption. In Ronald Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 457–473, Aarhus, Denmark, May 22–26, 2005. Springer, Heidelberg, Germany. 28

[SW14] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In David B. Shmoys, editor, *46th Annual ACM Symposium on Theory of Computing*, pages 475–484, New York, NY, USA, May 31 – June 3, 2014. ACM Press. 3, 4

[Zim15] Joe Zimmerman. How to obfuscate programs directly. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015, Part II*, volume 9057 of *Lecture Notes in Computer Science*, pages 439–467, Sofia, Bulgaria, April 26–30, 2015. Springer, Heidelberg, Germany. 3