

# On Extractability Obfuscation

Elette Boyle\*  
Cornell University  
ecb227@cornell.edu

Kai-Min Chung  
Academia Sinica  
kmchung@iis.sinica.edu.tw

Rafael Pass†  
Cornell University  
rafael@cs.cornell.edu

January 9, 2014

## Abstract

We initiate the study of *extractability obfuscation* (a.k.a. differing-inputs obfuscation), a notion first suggested by Barak *et al.* (JACM 2012): An extractability obfuscator  $e\mathcal{O}$  for a class of algorithms  $\mathcal{M}$  guarantees that if an efficient attacker  $\mathcal{A}$  can distinguish between obfuscations  $e\mathcal{O}(M_1), e\mathcal{O}(M_2)$  of two algorithms  $M_1, M_2 \in \mathcal{M}$ , then  $\mathcal{A}$  can efficiently recover (given  $M_1$  and  $M_2$ ) an input on which  $M_1$  and  $M_2$  provide different outputs.

- We rely on the recent candidate virtual black-box obfuscation constructions to provide candidate constructions of extractability obfuscators for  $NC^1$ ; next, following the blueprint of Garg *et al.* (FOCS 2013), we show how to bootstrap the obfuscator for  $NC^1$  to an obfuscator for all non-uniform polynomial-time Turing machines. In contrast to the construction of Garg *et al.*, which relies on indistinguishability obfuscation for  $NC^1$ , our construction enables succinctly obfuscating non-uniform *Turing machines* (as opposed to circuits), without turning running-time into description size.
- We introduce a new notion of *functional witness encryption*, which enables encrypting a message  $m$  with respect to an instance  $x$ , language  $L$ , and function  $f$ , such that anyone (and only those) who holds a witness  $w$  for  $x \in L$  can compute  $f(m, w)$  on the message and particular known witness. We show that functional witness encryption is, in fact, equivalent to extractability obfuscation.
- We demonstrate other applications of extractability extraction, including the first construction of fully (adaptive-message) indistinguishability-secure functional encryption for an unbounded number of key queries and unbounded message spaces.
- We finally relate indistinguishability obfuscation and extractability obfuscation and show special cases when indistinguishability obfuscation can be turned into extractability obfuscation.

---

\*Supported in part by AFOSR YIP Award FA9550-10-1-0093.

†Pass is supported in part by a Alfred P. Sloan Fellowship, Microsoft New Faculty Fellowship, NSF Award CNS-1217821, NSF CAREER Award CCF-0746990, NSF Award CCF-1214844, AFOSR YIP Award FA9550-10-1-0093, and DARPA and AFRL under contract FA8750-11-2-0211. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the US Government.

# 1 Introduction

**Obfuscation.** The goal of *program obfuscation* is to “scramble” a computer program, hiding its implementation details (making it hard to “reverse-engineer”), while preserving the functionality (i.e., input/output behavior) of the program. A first formal definition of such program obfuscation was provided by Hada [Had00]: roughly speaking, Hada’s definition—let us refer to it as *strongly virtual black-box*—is formalized using the simulation paradigm. It requires that anything an attacker can learn from the obfuscated code, could be simulated using just black-box access to the functionality.<sup>1</sup> Unfortunately, as noted by Hada, only learnable functionalities can satisfy such a strong notion of obfuscation: if the attacker simply outputs the code it is given, the simulator must be able to recover the code by simply querying the functionality and thus the functionality must be learnable.

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

Barak *et al.* [BGI<sup>+</sup>12] also suggested an avenue for circumventing these impossibility results:<sup>2</sup> introducing the notions of *indistinguishability* and “*differing-inputs*” obfuscation. Roughly speaking, an indistinguishability obfuscator  $i\mathcal{O}$  for a class of circuits  $\mathcal{C}$  guarantees that given any two *equivalent* circuits  $C_1$  and  $C_2$  (i.e., whose outputs agree on all inputs) from the class, obfuscations  $i\mathcal{O}(C_1)$  and  $i\mathcal{O}(C_2)$  of the circuits are indistinguishable. In a very recent breakthrough result, Garg, Gentry, Halevi, Raykova, Sahai, and Waters [GGH<sup>+</sup>13] provide the first candidate construction of indistinguishability obfuscators for all polynomial-size circuits. Additionally, Garg *et al.* [GGH<sup>+</sup>13] and even more recently, the elegant works of Sahai and Waters [SW13] and Hohenberger, Sahai and Waters [HSW13], demonstrate several beautiful (and surprising) applications of the indistinguishability obfuscation notion.

In this work, we initiate the study of the latter notion of obfuscation—“*differing-inputs*”, or as we call it, *extractability obfuscation*—whose security guarantees are at least as strong as indistinguishability obfuscation, but weaker than virtual black-box obfuscation. We demonstrate candidate constructions of such extractability obfuscators, and new applications.

**Extractability Obfuscation.** Roughly speaking, an extractability obfuscator  $e\mathcal{O}$  for a class of circuits  $\mathcal{C}$  guarantees that if an attacker  $\mathcal{A}$  can distinguish between obfuscations  $i\mathcal{O}(C_1), i\mathcal{O}(C_2)$  of two circuits  $C_1, C_2 \in \mathcal{C}$ , then  $\mathcal{A}$  can efficiently recover (given  $C_1$  and  $C_2$ ) a point  $x$  on which  $C_1$  and  $C_2$  differ: i.e.,  $C_1(x) \neq C_2(x)$ .<sup>3</sup> Note that if  $C_1$  and  $C_2$  are equivalent circuits, then no such input exists, thus requiring obfuscations of the circuits to be indistinguishable (and so extractability obfuscation implies indistinguishability obfuscation).

---

<sup>1</sup>Hada actually considered slight distributional weakening of this definition.

<sup>2</sup>Hada also suggested an approach for circumventing his impossibility result: he suggested sticking with a simulation-based definition, but instead restricting to particular classes of attacker. It is, however, not clear (to us) what reasonable classes of attackers are.

<sup>3</sup>Pedantically, our formalization is a slight variant of that of [BGI<sup>+</sup>12]; see Section 3 for details.

We may rely on the candidate obfuscator for  $NC^1$  of Brakerski and Rothblum [BR13] or Barak *et al.* [BGK<sup>+</sup>13] to obtain extractability obfuscation for the same class. We next demonstrate a bootstrapping theorem for extractability obfuscation, showing how to obtain extractability obfuscation for all polynomial-size circuits. Our transformation follows that of [GGH<sup>+</sup>13], but incurs a somewhat different analysis.

**Theorem 1.1** (Informally stated). *Assume the existence of an extractability obfuscator for  $NC^1$  and the existence of a (leveled) fully homomorphic encryption scheme with decryption in  $NC^1$  (implied, e.g., by Learning With Errors (LWE)). Then there exists an extractability obfuscation for  $P/poly$ .*

Relying on extractability obfuscation, however, has additional advantages: in particular, it allows us to achieve *succinctness* of the obfuscated program. Namely, we refer to an extractability (or indistinguishability) obfuscator as succinct if it can be used to obfuscate (non-uniform) Turing machines, while ensuring that the size of the obfuscated code preserves a polynomial relation to the size of the original Turing machine. In contrast, a non-succinct obfuscator may (and, indeed, the constructions of [GGH<sup>+</sup>13, BR13] do) turn running time into size: even if the original Turing machine has a short description, but a long running time, the obfuscated code will have a long description. To achieve succinctness, we are additionally required to rely on the existence of  $\mathbf{P}$ -certificates in the CRS model—namely, succinct non-interactive arguments for  $\mathbf{P}$ .<sup>4</sup>

**Theorem 1.2** (Informally stated). *Assume the existence of an extractability obfuscation for  $NC^1$ , the existence of a fully homomorphic encryption scheme with decryption in  $NC^1$  (implied, e.g., by LWE) and  $P$ -certificates (in the CRS model). Then there exists a succinct extractability obfuscation for  $P/poly$ .*

On a high level, our construction follows the one from [GGH<sup>+</sup>13] but (1) modifies it to deal with executions of Turing machines (by relying on an oblivious Turing machine), and more importantly (2) compresses “proofs” by using  $\mathbf{P}$ -certificates. We emphasize that this approach does *not* work in the setting of indistinguishability obfuscation. Intuitively, the reason for this is that  $\mathbf{P}$ -certificates of false statements *exist*, but are just hard to find; efficiently extracting such  $\mathbf{P}$ -certificates from a successful adversary is thus crucial (and enabled by the extractability property).

We next explore applications of extractability obfuscation.

**Functional Witness Encryption.** Consider the following scenario:

*You wish to encrypt the labels in a (huge) graph (e.g., names of people in a social network) so that no one can recover them, unless there is a clique in the graph of size, say, 100. Then, anyone (and only those) who knows such a clique should be able to recover the labels of the nodes in the identified clique (and only these nodes). Can this be done?*

The question is very related to the notion of *witness encryption*, recently introduced by Garg, Gentry, Sahai, and Waters [GGSW13]. Witness encryption makes it possible to encrypt the graph in such a way that anyone who finds any clique in the graph can recover the *whole* graph; if the graph does not contain any such cliques, the graph remains secret. The stronger notion of extractable witness encryption, introduced by Goldwasser, Kalai, Popa, Vaikuntanathan, and Zeldovich [GKP<sup>+</sup>13], further guarantees that the graph can only be decrypted by someone who actually

---

<sup>4</sup>Such certificates can be either based on knowledge-of-exponent type assumptions [BCCT13], or even on *falsifiable* assumptions [CLP13].

knows a clique. However, in contrast to existing notions, here we wish to reveal *only the labels associated with the particular known clique*.

More generally, we put forward the notion of *functional witness encryption (FWE)*. An FWE scheme enables one to encrypt a message  $m$  with respect to an  $NP$ -language  $L$ , instance  $x$  and a function  $f$ , such that anyone who has (and only those who have) a witness  $w$  for  $x \in L$  can recover  $f(m, w)$ . In the above example,  $m$  contains the labels of the whole graph,  $w$  is a clique, and  $f(m, w)$  are the labels of all the nodes in  $w$ . More precisely, our security definition requires that if you can tell apart encryptions of two messages  $m_0, m_1$ , then you must know a witness  $w$  for  $x \in L$  such that  $f(m_0, w) \neq f(m_1, w)$ .

We observe that general-purpose FWE and extractability obfuscation actually are equivalent (up to a simple transformation).

**Theorem 1.3** (Informally stated). *There exists a FWE for  $NP$  and every polynomial-size function  $f$  if and only if there exists an extractability obfuscator for every polynomial-size circuit.*

The idea is very simple: Given an extractability obfuscator  $e\mathcal{O}$ , an FWE encryption of the message  $m$  for the language  $L$ , instance  $x$  and function  $f$  is the obfuscation of the program that on input  $w$  outputs  $f(m, w)$  if  $w$  is a valid witness for  $x \in L$ . On the other hand, given a general-purpose FWE, to obfuscate a program  $\Pi$ , let  $f$  be the universal circuit that on input  $(\Pi, y)$  runs  $\Pi$  on input  $y$ , let  $L$  be the trivial language where every witness is valid, and output a FWE of the message  $\Pi$ —since every input  $y$  is a witness, this makes it possible to evaluate  $\Pi(y)$  on every  $y$ .

**Other Applications.** *Functional encryption* [BSW12, O’N10] enables the release of “targeted” secret keys  $sk_f$  that enable a user to recover  $f(m)$ , and *only*  $f(m)$ , given an encryption of  $m$ . It is known that strong simulation-based notions of security cannot be achieved if users can request an unbounded number of keys. In contrast, Garg *et al.* elegantly showed how to use indistinguishability obfuscation to satisfy an indistinguishability-based notion of functional encryption (roughly, that encryptions of any two messages  $m_0, m_1$  such that  $f(m_0) = f(m_1)$  for all the requested secret keys  $sk_f$  are indistinguishable). The main construction of Garg *et al.*, however, only achieves *selective-message* security, where the attacker must select the two message  $m_0, m_1$  to distinguish before the experiment begins (and it can request decryption keys  $sk_f$ ). Garg *et al.* observe that if they make subexponential-time security assumptions, use complexity leveraging, and consider a small (restricted) message space, then they can also achieve adaptive-message security.

We show how to use extractability obfuscation to directly achieve full adaptive-message security for any unbounded size message space (without relying on complexity leveraging).

The idea behind our scheme is as follows. Let the public key of the encryption scheme be the verification key for a signature scheme, and let the master secret key (needed to release secret keys  $sk_f$ ) be the signing key for the signature scheme. To encrypt a message  $m$ , obfuscate the program that on input  $f$  and a valid signature on  $f$  (with respect to the hardcoded public key) simply computes  $f(m)$ . The secret key  $sk_f$  for a function  $f$  is then simply the signature on  $f$ . (The high-level idea behind the construction is somewhat similar to the one used in [GKP<sup>+</sup>13], which used witness encryption in combination with signature schemes to obtain simulation-based FE for a *single* function  $f$ ; in contrast, we here focus on FE for an unbounded number of functions).

Proving that this construction works is somewhat subtle. In fact, to make the proof go through, we here require the signature scheme in use to be of a special *delegatable* kind—namely, we require the use of *functional signatures* [BGI13, BF13] (which can be constructed based on non-interactive zero-knowledge arguments of knowledge), which make it possible to delegate a signing key  $sk'$  that allows one to sign only messages satisfying some predicate. The delegation property is only used

in the security reduction and, roughly speaking, makes it possible to simulate key queries without harming security for the messages selected by the attacker.

**Theorem 1.4** (Informally stated). *Assume the existence of non-interactive zero-knowledge proofs of knowledge for NP and the existence of extractability obfuscators for polynomial-size circuits. Then there exists an (adaptive-message) indistinguishability-secure functional encryption scheme for arbitrary length messages.*

Another interesting feature of our approach is that it directly enables constructions of Hierarchical Functional Encryption (HiFE) (in analogy with Hierarchical Identity-Based encryption [HL02]), where the secret keys for functions  $f$  can be released in a hierarchical way (the top node can generate keys for subsidiary nodes, those nodes can generate keys for its subsidiaries etc.). To do this, simply modify the encryption algorithm to release the  $f(m)$  message in case you provide an appropriate chain of signatures that terminates with a signature on  $f$ .

**From Indistinguishability Obfuscation to Extractability Obfuscation.** A natural question is whether we can obtain extractability obfuscation from indistinguishability obfuscation. We address this question in two different settings: first directly in the context of obfuscation, and second in the language of FWE. (Recall that these two notions are equivalent when dealing with arbitrary circuits and arbitrary functions; however, when considering restricted function classes, there are interesting differences).

- We introduce a weaker form of extractability obfuscation, in which extraction is only required when the two circuits differ on only polynomially many inputs. We demonstrate that any indistinguishability obfuscation in fact implies weak extractability obfuscation.

**Theorem 1.5** (Informally stated). *Any indistinguishability obfuscator for  $P/\text{poly}$  is also a weak extractability obfuscator for  $P/\text{poly}$ .*

- Mirroring the definition of indistinguishability obfuscation, we may define a weaker notion of FWE—which we refer to as *indistinguishability FWE* (or iFWE)—which only requires that if  $f(m_0, w) = f(m_1, w)$  for *all* witnesses  $w$  for  $x \in L$ , then encryptions of  $m_0$  and  $m_1$  are indistinguishable (in contrast, the stronger notion requires that if you can distinguish between encryptions of  $m_0$  and  $m_1$  you must know a witness on which they differ). It follows that iFWE for languages in NP and functions in  $P/\text{poly}$  is equivalent to indistinguishability obfuscation for  $P/\text{poly}$ , up to a simple transformation. We show that if restricting to languages with polynomially many witnesses, it is possible to turn any iFWE to an FWE.

**Theorem 1.6** (Informally stated). *Assume there exists indistinguishability FWE for every NP language with polynomially many witnesses, and the function  $f$ . Then for every language  $L$  in NP with polynomially many witnesses, there exists an FWE for  $L$  and  $f$ .*

Our proof relies on a local list-decoding algorithm for a large-alphabet Hadamard code due to Goldreich, Rubinfeld and Sudan [GRS00].

Theorems 1.5 and 1.6 are incomparable in that Theorem 1.5 begins with a stronger assumption and yields a stronger conclusion. More precisely, if one begins with iFWE supporting all languages in NP and functions in  $P/\text{poly}$ , then the equivalence between indistinguishability (respectively, standard) FWE and indistinguishability (resp., extractability) obfuscation, in conjunction with the transformation of Theorem 1.5, yields a *stronger* outcome in the setting of FWE than Theorem 1.6: Namely, a form of FWE where (extraction) security holds as long as the function  $M(m, w)$  is

not “too sensitive” to  $m$ : i.e., if for any two messages  $m_0, m_1$  there are only polynomially many witnesses  $w$  for which  $M(m_0, w) \neq M(m_1, w)$ . This captures, for example, functions  $M$  that only rarely output nonzero values. Going back to the example of encrypting data  $m$  associated with nodes of a social network, we could then allow someone holding clique  $w$  to learn whether the nodes in this clique satisfy some chosen rare property (e.g., contains someone with a rare disease, all have the same birthday, etc). Indeed, while there may be many cliques (corresponding to several, even exponentially many, witnesses  $w$ ), it will be the case that  $M(m, w)$  is almost always 0, for all but polynomially many  $w$ .

On the other hand, Theorem 1.6 also provides implications of iFWE for restricted function classes. In particular, Theorem 1.6 gives a method for transforming indistinguishability FWE for the trivial function  $f(m, w) = m$  to FWE for the same function  $f$ . It is easy to see that indistinguishability FWE for this particular  $f$  is equivalent to the notion of witness encryption [GGSW13], and FWE for the same  $f$  is equivalent to the notion of extractable witness encryption of [GKP<sup>+</sup>13]. Theorem 1.6 thus shows how to turn witness encryption to extractable witness encryption for the case of languages with polynomially many witness.

Finally, we leave open whether there are corresponding transformations from indistinguishability obfuscation in the case of many disagreeing inputs, and iFWE in the case of many witnesses. In the latter setting, this is interesting even for the special case of witness encryption (i.e., the function  $f(m, w) = m$ ).

## 1.1 Overview of the Paper

In Section 2, we present definitions and notation for some of the tools used in the paper. In Section 3, we introduce the notion of extractability obfuscation and present a bootstrapping transformation from any extractability obfuscator for  $NC^1$  to one for all of  $P/\text{poly}$ . In Section 4, we define functional witness encryption (FWE), and show an equivalence between FWE and extractability obfuscation. In Section 5, we describe an application of extractability obfuscation, in achieving indistinguishability functional encryption with unbounded-size message space. In Section 6, we explore the relationship between indistinguishability and extractability obfuscation, providing transformations from the former to the latter in special cases.

## 2 Preliminaries

### 2.1 Fully Homomorphic Encryption

A fully homomorphic encryption scheme  $\mathcal{E} = (\text{Gen}_{\text{FHE}}, \text{Enc}_{\text{FHE}}, \text{Dec}_{\text{FHE}}, \text{Eval}_{\text{FHE}})$  is a public-key encryption scheme that associates with an additional polynomial-time algorithm  $\text{Eval}_{\text{FHE}}$ , which takes as input a public key  $\text{pk}$ , a ciphertext  $c = \text{Enc}_{\text{FHE}}(\text{pk}, m)$  and a circuit  $C$ , and outputs, a new ciphertext  $c' = \text{Eval}_{\text{FHE}}(\text{pk}, c, C)$ , such that  $\text{Dec}_{\text{FHE}}(\text{sk}, c') = C(m)$ , where  $\text{sk}$  is the secret key corresponding to the public key  $\text{pk}$ . Formally, we require  $\mathcal{E}$  to have the following correctness property:

**Definition 2.1** (FHE correctness). There exists a negligible function  $\nu(k)$  such that

$$\Pr_{\text{pk}, \text{sk} \leftarrow \text{Gen}(1^k)} \left[ \begin{array}{l} \forall \text{ ciphertexts } c_1, \dots, c_n \text{ s.t. } c_i \leftarrow \text{Enc}_{\text{pk}}(b_i), \\ \forall \text{ poly-size circuits } f : \{0, 1\}^n \rightarrow \{0, 1\} \\ \text{Dec}_{\text{sk}}(\text{Eval}_{\text{pk}}(f, c_1, \dots, c_n)) = f(b_1, \dots, b_n), \end{array} \right] \geq 1 - \nu(k).$$

It is required that the size of  $c' = \text{Eval}_{\text{FHE}}(\text{pk}, \text{Enc}_{\text{FHE}}(\text{pk}, m), C)$  depends polynomially on the security parameter and the length of  $C(m)$ , but is otherwise independent of the size of the circuit  $C$ . For security, we simply require that  $\mathcal{E}$  is semantically secure. We also require that  $\text{Eval}$  is deterministic, and that the decryption circuit  $\text{Dec}_{\text{sk}}(\cdot)$  is in  $NC^1$ . Most known schemes satisfy these properties. Since the breakthrough of Gentry [Gen09], several fully homomorphic encryption schemes have been constructed with improved efficiency and based on more standard assumptions such as LWE (Learning With Errors) (e.g., [BV11, BGV11, GSW13, BV13]).

**Remark 2.2** (Homomorphic evaluation of Turing machines). As part of our extractability obfuscation construction for general Turing machines (TM), we require the homomorphic evaluation of an *oblivious* Turing machine with known runtime. Recall that a Turing machine is said to be oblivious if its tape movements are independent of its input. The desired homomorphic evaluation is done as follows.

Suppose  $\hat{x} = (\hat{x}_1, \hat{x}_2, \dots, \hat{x}_k)$  is an FHE encryption of plaintext message  $x$  (where  $\hat{x}_\ell$  encrypts the  $\ell$ th position of  $x$ ),  $\hat{a} = (\hat{a}_1, \hat{a}_2, \dots)$  an FHE encryption of the tape values,  $\hat{s}$  an FHE ciphertext of the current state, and  $M$  an oblivious Turing machine terminating on all inputs within  $t$  steps. More specifically, a description of  $M$  consists of an initial state  $s$  and description of a transition circuit,  $C_M$ . In each step  $i = 1, \dots, t$  of evaluation,  $M$  accesses some fixed position  $\text{pos}_{\text{input}}(i)$  of the input, fixed position  $\text{pos}_{\text{tape}}(i)$  of the tape (extending straightforwardly to the multi-tape setting), and the current value of the state, and evaluates  $C_M$  on these values.

Homomorphic evaluation of  $M$  on the encrypted input  $\hat{x}$  then takes place in  $t$  steps: In each step  $i$ , the transition circuit  $C_M$  of  $M$  is homomorphically evaluated on the ciphertexts  $\hat{x}_{\text{pos}_{\text{input}}}$ ,  $\hat{a}_{\text{pos}_{\text{tape}}}$ , and  $\hat{s}$ , yielding updated values for these ciphertexts. The updated state ciphertext  $\hat{s}$  resulting after  $t$  steps is the desired output ciphertext. Note that obliviousness of the Turing machine is crucial for this efficient method of homomorphic evaluation, as any input-dependent choices for the head location would only be available to an evaluator in encrypted form.

Overall, homomorphic evaluation of  $M$  takes time  $O(t(k) \cdot \text{poly}(k))$ , and can be described in space  $O(|M| \cdot \text{poly}(k))$ .

## 2.2 (Indistinguishability) Functional Encryption

A functional encryption scheme [BSW12, O’N10] enables the release of “targeted” secret keys that enable a user to recover  $f(m)$ —and only  $f(m)$ —given an encryption of  $m$ . In this work, we will consider the indistinguishability notion of security for functional encryption. Roughly, such a scheme is said to be secure if an adversary who requests and learns secret keys  $\text{sk}_f$  for a collection of functions  $f$  cannot distinguish encryptions of any two messages  $m_0, m_1$  for which  $f(m_0) = f(m_1)$  for every requested  $f$ .

**Definition 2.3** (Functional Encryption). [BSW12, O’N10] A *functional encryption scheme* for a class of functions  $F = \mathcal{F}(k)$  over message space  $\mathcal{M} = \mathcal{M}_k$  consists of four algorithms  $\mathcal{FE} = (\text{Setup}, \text{KeyGen}, \text{Enc}, \text{Dec})$  with syntax

- $\text{Setup}(1^k)$ : on input the security parameter  $1^k$ ,  $\text{Setup}$  outputs public parameters  $\text{pp}$  and a master secret key  $\text{msk}$ .
- $\text{KeyGen}(\text{msk}, f)$ : on input the master secret key  $\text{msk}$  and function description  $f \in \mathcal{F}$ ,  $\text{KeyGen}$  outputs a secret key  $\text{sk}_f$ .
- $\text{Enc}(\text{pp}, m)$ : on input public parameters  $\text{pp}$  and message  $m$ ,  $\text{Enc}$  outputs a ciphertext  $c$ .
- $\text{Dec}(\text{sk}_f, c)$ : on input a secret key  $\text{sk}_f$  and ciphertext  $c$  output an evaluated plaintext  $m'$  (allegedly corresponding to  $f(m)$ ).

satisfying the following properties:

- **Correctness:** For every message  $m \in \mathcal{M}$  and function  $f \in \mathcal{F}$ , there exists a negligible function  $\mu(k)$  for which

$$\Pr \left[ (\text{pp}, \text{msk}) \leftarrow \text{Setup}(1^k); \text{sk}_f \leftarrow \text{KeyGen}(\text{msk}, f) : \text{Dec}(\text{sk}_f, \text{Enc}(\text{pp}, m)) \neq f(m) \right] \leq \mu(k).$$

- **Indistinguishability Security:** The advantage of any PPT adversary  $\mathcal{A}$  in the following challenge is negligible in  $k$ :
  1. Setup: The challenger samples  $(\text{pp}, \text{msk}) \leftarrow \text{Setup}(1^k)$  and gives the public parameters  $\text{pp}$  to  $\mathcal{A}$ .
  2. Key Queries:  $\mathcal{A}$  adaptively submits queries  $f_i \in \mathcal{F}$  and is given  $\text{sk}_{f_i} \leftarrow \text{KeyGen}(\text{msk}, f_i)$  for each query. This step can be repeated any polynomial number of times.
  3. Challenge:  $\mathcal{A}$  submits two messages  $m_0, m_1 \in \mathcal{M}$  for which  $f_i(m_0) = f_i(m_1)$  for every function  $f_i$  queried in the key query phase, and is given a challenge ciphertext  $c \leftarrow \text{Enc}(\text{pp}, m_b)$  for a randomly selected bit  $b \leftarrow \{0, 1\}$ .
  4. Additional Key Queries:  $\mathcal{A}$  adaptively submits queries  $f_i \in \mathcal{F}$ , subject to the restriction that  $f_i(m_0) = f_i(m_1)$ . For each such query,  $\mathcal{A}$  is given  $\text{sk}_{f_i} \leftarrow \text{KeyGen}(\text{msk}, f_i)$ . This step can be repeated any polynomial number of times.
  5. Guess:  $\mathcal{A}$  outputs a bit  $b' \in \{0, 1\}$ .

The advantage of  $\mathcal{A}$  in the game above is defined to be  $|\Pr[b' = b] - \frac{1}{2}|$ .

### 2.3 P-Certificates

**P**-Certificates are a succinct argument system for **P**. We present **P** certificates in the CRS model.

Consider the following canonical languages for **P**: for every constant  $c \in \mathbb{N}$ , let  $L_c = \{(M, x, y) : M(x) = y \text{ within } |x|^c \text{ steps}\}$ . Let  $T_M(x)$  denote the running time of  $M$  on input  $x$ .

**Definition 2.4 (P-certificates).** [CLP12] A tuple of probabilistic interactive Turing machines  $(\text{CRSGen}_{\text{cert}}, P_{\text{cert}}, V_{\text{cert}})$  is a **P-certificate system** in the CRS model if there exist polynomials  $g_P, g_V, \ell$  such that the following hold:

- **Efficient Verification:** On input  $\text{crs} \leftarrow \text{CRSGen}(1^k)$ ,  $c \geq 1$ , and a statement  $q = (M, x, y) \in L_c$ , and  $\pi \in \{0, 1\}^*$ ,  $V_{\text{cert}}$  runs in time at most  $g_V(k + |q|)$ .
- **Completeness by a Relatively Efficient Prover:** For every  $c, d \in \mathbb{N}$ , there exists a negligible function  $\mu$  such that for every  $k \in \mathbb{N}$  and every  $q = (M, x, y) \in L_c$  such that  $|q| \leq k^d$ ,

$$\Pr[\text{crs} \leftarrow \text{CRSGen}(1^k); \pi \leftarrow P_{\text{cert}}(\text{crs}, c, q) : V_{\text{cert}}(\text{crs}, c, q, \pi) = 1] \geq 1 - \mu(k).$$

Furthermore,  $P_{\text{cert}}$  on input  $(\text{crs}, c, q)$  outputs a certificate of length  $\ell(k)$  in time bounded by  $g_P(k + |M| + T_M(x))$ .

- **Soundness:** For every  $c \in \mathbb{N}$ , and every (not necessarily uniform) PPT  $P^*$ , there exists a negligible function  $\mu$  such that for every  $k \in \mathbb{N}$ ,

$$\Pr[\text{crs} \leftarrow \text{CRSGen}(1^k); (q, \pi) \leftarrow P^*(\text{crs}, c) : V_{\text{cert}}(\text{crs}, c, q, \pi) = 1 \wedge q \notin L_c] \leq \mu(k).$$

**P** certificates are directly implied by any publicly-verifiable succinct non-interactive argument system (SNARG) for **P**. In particular, we have the following.



**Theorem 2.5.** *Assuming that Micali’s CS proof [Mic00] is sound, or assuming the existence of publicly-verifiable fully succinct SNARG system for  $\mathbf{P}$  [BCCT13] (which in turn can be based on any publicly-verifiable SNARG [Gro10, Lip12, GGPR13, BCT<sup>+</sup>13]), then there exists a  $\mathbf{P}$ -certificate system in the CRS model.*

It was shown by Chung et al. [CLP12] that  $\mathbf{P}$ -certificates can be based on *falsifiable* assumptions [Nao03].

## 2.4 Functional Signatures

In a functional signature scheme, in addition to a *master signing key* that can be used to sign any message, there are secondary *signing keys for functions  $f$*  (called  $\text{sk}_f$ ), which allow one to sign any message in the range of  $f$ .<sup>5</sup> We present the definition as considered in [BGI13]. A similar notion was presented in [BF13] for the special case of functions corresponding to predicates.

**Definition 2.6** (Functional Signatures). [BGI13] A *functional signature scheme* for a message space  $\mathcal{M}$ , and function family  $\mathcal{F} = \{f : \mathcal{D}_f \rightarrow \mathcal{M}\}$  consists of algorithms (FS.Setup, FS.KeyGen, FS.Sign, FS.Verify):

- FS.Setup( $1^k$ )  $\rightarrow$  (msk, mvk): the setup algorithm takes as input the security parameter and outputs the master signing key and master verification key.
- FS.KeyGen(msk,  $f$ )  $\rightarrow$   $\text{sk}_f$ : the KeyGen algorithm takes as input the master signing key and a function  $f \in \mathcal{F}$  (represented as a circuit), and outputs a signing key for  $f$ .
- FS.Sign( $f, \text{sk}_f, m$ )  $\rightarrow$  ( $f(m), \sigma$ ): the signing algorithm takes as input the signing key for a function  $f \in \mathcal{F}$  and an input  $m \in \mathcal{D}_f$ , and outputs  $f(m)$  and a signature of  $f(m)$ .
- FS.Verify(mvk,  $m^*, \sigma$ )  $\rightarrow$   $\{0, 1\}$ : the verification algorithm takes as input the master verification key mvk, a message  $m$  and a signature  $\sigma$ , and outputs 1 if the signature is valid.

We say that FS is a functional signature scheme for *unbounded-length messages* if  $\mathcal{M} = \{0, 1\}^*$ .

We require the following conditions to hold:

**Corectness:**

$$\forall f \in \mathcal{F}, \forall m \in \mathcal{D}_f, (\text{msk}, \text{mvk}) \leftarrow \text{FS.Setup}(1^k), \text{sk}_f \leftarrow \text{FS.KeyGen}(\text{msk}, f), (m^*, \sigma) \leftarrow \text{FS.Sign}(f, \text{sk}_f, m),$$

$$\text{FS.Verify}(\text{mvk}, m^*, \sigma) = 1.$$

**Unforgeability:**

The scheme is unforgeable if the advantage of any PPT algorithm A in the following game is negligible:

- The challenger generates  $(\text{msk}, \text{mvk}) \leftarrow \text{FS.Setup}(1^k)$ , and gives mvk to A
- The adversary is allowed to query a key generation oracle  $\mathcal{O}_{\text{key}}$ , and a signing oracle  $\mathcal{O}_{\text{sign}}$ , that share a dictionary indexed by tuples  $(f, i)$ , whose entries are signing keys:  $\text{sk}_f^i \leftarrow \text{FS.KeyGen}(\text{msk}, f)$ . This dictionary keeps track of the keys that have been previously generated during the unforgeability game. The oracles are defined as follows :
  - $\mathcal{O}_{\text{key}}(f, i)$  :

---

<sup>5</sup>Note that this includes as a special case signing permissions defined by predicates, e.g. by considering the function  $f_P(x) = x$  if  $P(x) = 1$  and  $= \perp$  if  $P(x) = 0$ .

- \* if there exists an entry for the key  $(f, i)$  in the dictionary, then output the corresponding value,  $\text{sk}_f^i$ .
- \* otherwise, sample a fresh key  $\text{sk}_f^i \leftarrow \text{FS.KeyGen}(\text{msk}, f)$ , add an entry  $(f, i) \rightarrow \text{sk}_f^i$  to the dictionary, and output  $\text{sk}_f^i$
- $\text{O}_{\text{sign}}(f, i, m)$ :
  - \* if there exists an entry for the key  $(f, i)$  in the dictionary, then generate a signature on  $f(m)$  using this key:  $\sigma \leftarrow \text{FS.Sign}(f, \text{sk}_f^i, m)$ .
  - \* otherwise, sample a fresh key  $\text{sk}_f^i \leftarrow \text{FS.KeyGen}(\text{msk}, f)$ , add an entry  $(f, i) \rightarrow \text{sk}_f^i$  to the dictionary, and generate a signature on  $f(m)$  using this key:  $\sigma \leftarrow \text{FS.Sign}(f, \text{sk}_f^i, m)$ .
- The adversary wins if it can produce  $(m^*, \sigma)$  such that
  - $\text{FS.Verify}(\text{mvk}, m^*, \sigma) = 1$ .
  - there does not exist  $m$  such that  $m^* = f(m)$  for any  $f$  which was sent as a query to the  $\text{O}_{\text{key}}$  oracle.
  - there does not exist a  $(f, m)$  pair such that  $(f, m)$  was a query to the  $\text{O}_{\text{sign}}$  oracle and  $m^* = f(m)$ .

**Function privacy:**

Intuitively, we require the distribution of signatures on a message  $m'$  generated via different keys  $\text{sk}_f$  to be computationally indistinguishable, *even given the secret keys and master signing key*. Namely, the advantage of any PPT adversary in the following game is negligible:

- The challenger honestly generates a key pair  $(\text{mvk}, \text{msk}) \leftarrow \text{FS.Setup}(1^k)$  and gives both values to the adversary. (Note wlog this includes the randomness used in generation).
- The adversary chooses a function  $f_0$  and receives an (honestly generated) secret key  $\text{sk}_{f_0} \leftarrow \text{FS.KeyGen}(\text{msk}, f_0)$ .
- The adversary chooses a second function  $f_1$  for which  $|f_0| = |f_1|$  (where padding can be used if there is a known upper bound) and receives an (honestly generated) secret key  $\text{sk}_{f_1} \leftarrow \text{FS.KeyGen}(\text{msk}, f_1)$ .
- The adversary chooses a pair of values  $m_0, m_1$  for which  $|m_0| = |m_1|$  and  $f_0(m_0) = f_1(m_1)$ .
- The challenger selects a random bit  $b \leftarrow \{0, 1\}$  and generates a signature on the image message  $m' = f_0(m_0) = f_1(m_1)$  *using secret key*  $\text{sk}_{f_b}$ , and gives the resulting signature  $\sigma \leftarrow \text{FS.Sign}(\text{sk}_{f_b}, m_b)$  to the adversary.
- The adversary outputs a bit  $b'$ , and wins the game if  $b' = b$ .

**Succinctness:**

The size of a signature  $\sigma \leftarrow \text{FS.Sign}(\text{sk}_f, m)$  is bounded by a polynomial in the security parameter  $k$ , and the size of the output  $|f(m)|$ . In particular, it is independent of  $|m|$ , the size of the input to the function, and  $|f|$ , the size of a description of the function  $f$ .

**Theorem 2.7** ([BGI13, BF13]). *Given any NIZK argument of knowledge for NP, there exists a functional signature scheme for unbounded-length messages and functions, satisfying unforgeability and function privacy. Assuming a succinct non-interactive argument of knowledge (SNARK) for NP, the corresponding functional signature scheme is also succinct.*

We also remark that a straightforward modification of these constructions also yield an analogous result for *Turing machines*  $f$ .

### 3 Extractability Obfuscation

We now present and study the notion of *extractability obfuscation*, which is a slight relaxation of “differing-inputs obfuscation” introduced in [BGI<sup>+</sup>12]. Intuitively, such an obfuscator has the property that if a PPT adversary can distinguish between obfuscations of two programs  $M_0, M_1$ , then he must “know” an input on which they differ.

For simplicity, we present and refer primarily to a security definition with respect to worst-case auxiliary input. It has become increasingly evident, however, that the achievability of extraction-type objects may depend greatly on the level of auxiliary-input resilience demanded, and for the case of obfuscation such a worst-case guarantee may be too strong to hope for (e.g., [HT98, BCPR13, BP13, GGHW13]). We thus also present two weaker distributional variants, in which the extraction property is only required to hold for specific distributions of auxiliary input and/or programs in the class  $\mathcal{M}_k$ . We remark that these weaker notions suffice for our applications, and that no negative results for the application-required auxiliary input distributions are known.<sup>6</sup> We also present security with respect to non-uniform adversaries; analogous versions can be considered for the case of uniform adversaries (in which case the corresponding extractor is also required to be uniform).

**Definition 3.1** (Extractability Obfuscator – Worst-Case Inputs). (Variant of [BGI<sup>+</sup>12]<sup>7</sup>) A uniform PPT machine  $e\mathcal{O}$  is an *extractability obfuscator* (w.r.t. worst-case inputs) for a class of Turing machines  $\{\mathcal{M}_k\}_{k \in \mathbb{N}}$  if the following conditions are satisfied:

- **Correctness:** There exists a negligible function  $\text{negl}(k)$  such that for every security parameter  $k \in \mathbb{N}$ , for all  $M \in \mathcal{M}_k$ , for all inputs  $x$ , we have

$$\Pr[M' \leftarrow e\mathcal{O}(1^k, M) : M'(x) = M(x)] = 1 - \text{negl}(k).$$

- **Security:** For every non-uniform PPT adversary  $\mathcal{A}$  and polynomial  $p(k)$ , there exists a non-uniform PPT extractor  $E$  and polynomial  $q(k)$  such that the following holds. For every  $k \in \mathbb{N}$ , every pair of Turing machines  $M_0, M_1 \in \mathcal{M}_k$ , and every auxiliary input  $z$ ,

$$\Pr \left[ b \leftarrow \{0, 1\}; M' \leftarrow e\mathcal{O}(1^k, M_b) : \mathcal{A}(1^k, M', M_0, M_1, z) = b \right] \geq \frac{1}{2} + \frac{1}{p(k)} \quad (1)$$

$$\implies \Pr \left[ w \leftarrow E(1^k, M_0, M_1, z) : M_0(w) \neq M_1(w) \right] \geq \frac{1}{q(k)}. \quad (2)$$

We now present a distributional variant, where instead of requiring extraction to hold with respect to every program pair  $M_0, M_1 \in \mathcal{M}_k$  and auxiliary input  $z \in \{0, 1\}^*$ , we consider a distribution  $\mathcal{D}$  that samples  $(M_0, M_1, z) \leftarrow \mathcal{D}$  and require that for this distribution  $\mathcal{D}$ , there exists an extractor such that the above extraction condition holds.

**Definition 3.2** (Extractability Obfuscator w.r.t. Distribution  $\mathcal{D}$ ). A uniform PPT machine  $e\mathcal{O}$  is an *extractability obfuscator w.r.t. distribution  $\mathcal{D}$*  over  $\mathcal{M}_k \times \mathcal{M}_k \times \{0, 1\}^*$  if it satisfies the correctness property from Definition 3.1, in addition to the following (weaker) extraction property:

<sup>6</sup>Indeed, currently existing evidence of impossibility makes use of complex auxiliary input distributions, formed by obfuscating particular programs.

<sup>7</sup>Formally, our notion of extractability obfuscation departs from differing-inputs obfuscation of [BGI<sup>+</sup>12] in three ways: First, we consider a setting in which the adversary (and extractor) may receive auxiliary input. Second, [BGI<sup>+</sup>12] require the extractor  $E$  to extract a differing input for  $M_0, M_1$  given *any* pair of programs  $M'_0, M'_1$  evaluating equivalent functions. Finally, [BGI<sup>+</sup>12] consider also adversaries who distinguish with negligible advantage  $\epsilon(k)$ , and require that extraction still succeeds in this setting, but within time polynomial in  $1/\epsilon$ . In contrast, we restrict our attention only to adversaries who succeed with noticeable advantage.

- **Security:** For every non-uniform PPT adversary  $\mathcal{A}$  and polynomial  $p(k)$ , there exists a non-uniform PPT extractor  $E$  and polynomial  $q(k)$  such that for every  $k \in \mathbb{N}$ , it holds with overwhelming probability over  $(M_0, M_1, z) \leftarrow \mathcal{D}$  that

$$\begin{aligned} \Pr \left[ b \leftarrow \{0, 1\}; M' \leftarrow e\mathcal{O}(1^k, M_b) : \mathcal{A}(1^k, M', M_0, M_1, z) = b \right] &\geq \frac{1}{2} + \frac{1}{p(k)} \\ \implies \Pr \left[ w \leftarrow E(1^k, M_0, M_1, z) : M_0(w) \neq M_1(w) \right] &\geq \frac{1}{q(k)}. \end{aligned}$$

Finally, we consider security against general distributional auxiliary input, in which the above extraction property holds for every efficiently samplable distribution  $\mathcal{D}$ .

**Definition 3.3** (Extractability Obfuscator w.r.t. Distributional Auxiliary Input). A uniform PPT machine  $e\mathcal{O}$  is an *extractability obfuscator w.r.t. (general) distributional auxiliary input* for the class of Turing machines  $\{\mathcal{M}_k\}_{k \in \mathbb{N}}$  if it satisfies the correctness property from Definition 3.1 in addition to the following extraction property:

- **Security:** For every non-uniform PPT adversary  $\mathcal{A}$ , polynomial  $p(k)$ , and efficiently samplable distribution  $\mathcal{D}$  over  $\mathcal{M}_k \times \mathcal{M}_k \times \{0, 1\}^*$  there exists a non-uniform PPT extractor  $E$  and polynomial  $q(k)$  such that for every  $k \in \mathbb{N}$ , it holds with overwhelming probability over  $(M_0, M_1, z) \leftarrow \mathcal{D}$  that

$$\begin{aligned} \Pr \left[ b \leftarrow \{0, 1\}; M' \leftarrow e\mathcal{O}(1^k, M_b) : \mathcal{A}(1^k, M', M_0, M_1, z) = b \right] &\geq \frac{1}{2} + \frac{1}{p(k)} \\ \implies \Pr \left[ w \leftarrow E(1^k, M_0, M_1, z) : M_0(w) \neq M_1(w) \right] &\geq \frac{1}{q(k)}. \end{aligned}$$

Note that the extractor  $E$  in this definition may depend not only on the adversary, but also on the distribution  $\mathcal{D}$ .

We contrast the notion of extractability obfuscation with that of *indistinguishability obfuscation*:

**Definition 3.4** (Indistinguishability Obfuscator). [BGI<sup>+</sup>12] A uniform PPT machine  $i\mathcal{O}$  is an *indistinguishability obfuscator* for a class of circuits  $\{\mathcal{C}_k\}$  if  $i\mathcal{O}$  satisfies the Correctness and Security properties as in Definition 3.1 (for circuit class  $\{\mathcal{C}_k\}$  and circuits  $C_0, C_1$  in the place of Turing machines), except with Line (2) replaced with the following:

$$\implies \exists w : C_0(w) \neq C_1(w). \tag{2'}$$

Note that any *extractability* obfuscator is also directly an *indistinguishability* obfuscator, since existence of an efficient extraction algorithm  $E$  finding desired distinguishing input  $w$  as in (2) in particular implies that such an input exists, as in (2').

**Remark 3.5.** Note that in the definition of extractability obfuscation, the extractor is given access to the programs  $M_0, M_1$ . One could consider an even stronger notion of obfuscation, in which the extractor is given only black-box access to the two programs. As we show in Appendix A, however, achieving general-purpose obfuscation satisfying this stronger extractability notion is impossible.

We now present specific definitions of extractability obfuscators for special classes of Turing machines.

**Definition 3.6** (Extractability Obfuscator for  $NC^1$ ). A uniform PPT machine  $e\mathcal{O}_{NC^1}$  is called an *extractability obfuscator for  $NC^1$*  if for constants  $c \in \mathbb{N}$ , the following holds: Let  $\mathcal{M}_k$  be the class of Turing machines corresponding to the class of circuits of depth at most  $c \log k$  and size at most  $k$ . Then  $e\mathcal{O}(c, \cdot, \cdot)$  is an extractability obfuscator for the class  $\{\mathcal{M}_k\}$ .

**Definition 3.7** (Extractability Obfuscator for TM). A uniform PPT machine  $e\mathcal{O}_{TM}$  is called an *extractability obfuscator for the class TM of polynomial-size Turing machines* if it satisfies the following. For each  $k$ , let  $\mathcal{M}_k$  be the class of Turing machines  $\Pi$  containing a description of a Turing machine  $M$  of size bounded by  $k$ , such that  $\Pi$  takes two inputs,  $(t, x)$ , with  $|t| = k$ , and the output of  $\Pi(t, x)$  is defined to be the the output of running the Turing machine  $M(x)$  for  $t$  steps. Then  $e\mathcal{O}_{TM}$  is an extractability obfuscator for  $\{\mathcal{M}_k\}$ .

Applying the properties of extractability obfuscation to this class of Turing machines  $\{\mathcal{M}_k\}$  implies that for programs  $\Pi_0, \Pi_1 \in \mathcal{M}_k$  defined above (corresponding to underlying size- $k$  Turing machines  $M_0, M_1$ ), efficiently distinguishing between obfuscations of  $\Pi_0$  and  $\Pi_1$  implies that one can efficiently extract an input *pair*  $(t', x')$  for which  $\Pi_0(t', x') \neq \Pi_1(t', x')$ . In particular, either  $M_0(x') \neq M_1(x')$  or  $\text{Runtime}(M_0, x') \neq \text{Runtime}(M_1, x)$ . Thus, if restricting attention to a subclass of  $\mathcal{M}_k$  for which each pair of programs satisfies  $\text{Runtime}(M_0, x) = \text{Runtime}(M_1, x)$  for each input  $x$ , then “standard” extraction is guaranteed (i.e., such that the extracted input contains  $x'$  satisfying  $M_0(x') \neq M_1(x')$ ), while achieving input-specific runtime of the obfuscated program. (Indeed, for an input  $x$  of unknown runtime, one simply executes the obfuscated program  $\tilde{\Pi}$  sequentially with increasing time bounds  $t = k, k^2, k^{2^2}, k^{2^3}, \dots$  until a non- $\perp$  output is received). If restricting to a subclass  $\mathcal{M}_k$  that has a polynomial runtime bound  $t(k)$ , then “standard” extraction can be guaranteed by simply defining  $\text{Runtime}(M, x) = t(k)$  for every  $M \in \mathcal{M}_k$  and every input  $x$ .

In the sequel, when referring to an extractability obfuscation of a Turing machine  $M$ , we will implicitly mean the related program  $\Pi_M$  as above, but will suppress notation of the additional input  $t$ .

**Definition 3.8** (Extractability Obfuscator for Bounded-Input TM). A uniform PPT machine  $e\mathcal{O}_{BI}$  is called an extractability obfuscator for bounded-input Turing machines if it satisfies the following. For each  $k$  and polynomial  $\ell(k)$ , let  $\mathcal{M}_k^\ell$  be the class of Turing machines  $\Pi$  as in Definition 3.7, but where the inputs  $(t, x)$  of  $\Pi$  are limited by  $|t| = k$  and  $|x| \leq \ell(k)$ . Then there exists a polynomial  $p_s(k)$  for which  $e\mathcal{O}_{BI}$  is an extractability obfuscator for  $\{\mathcal{M}_k^\ell\}$ , and for every  $k \in \mathbb{N}$ , and every  $M \in \mathcal{M}_k^\ell$ , it holds that the obfuscation  $M' \leftarrow e\mathcal{O}_{BI}(1^k, M, \ell(k))$  has size bounded by  $p_s(\ell(k), k)$ .

### 3.1 Extractability Obfuscation for $NC^1$

In this work, we build upon the existence of any extractability obfuscator for  $NC^1$ .

**Assumption 3.9** ( $NC^1$  Extractability Obfuscator). *There exists a secure extractability obfuscator  $e\mathcal{O}_{NC^1}$  for  $NC^1$ , as in Definition 3.6*

In particular, this assumption can be instantiated using the candidate obfuscator for  $NC^1$  given by Brakerski and Rothblum [BR13] or Barak et al. [BGK<sup>+</sup>13]. These works achieve (stronger) *virtual black-box* security within an idealized model, based on certain assumptions. We refer the reader to [BR13, BGK<sup>+</sup>13] for more details.

### 3.2 Amplifying to General Polynomial-Sized Turing Machines

In this section, we demonstrate how to bootstrap from an extractability obfuscator for  $NC^1$  to one for *all* (bounded-input) Turing machines with a polynomial-sized description, by use of fully

homomorphic encryption (FHE), in conjunction with a  $\mathbf{P}$ -certificate system (a succinct argument system for statements in  $\mathbf{P}$ ).<sup>8</sup> Our construction provides also two corollaries. Relaxing our assumptions, by using *leveled* FHE, and removing  $\mathbf{P}$ -certificates, we achieve extractability obfuscation for polynomial-size circuits. And strengthening our assumption, replacing  $\mathbf{P}$ -certificates with succinct non-interactive arguments of knowledge (SNARKs), yields extractability obfuscation for all polynomial-size Turing machines. Our construction follows the analogous amplification transformation of Garg et. al [GGH<sup>+</sup>13] in the (weaker) setting of indistinguishability obfuscation.

At a high level, the transformation works as follows. An obfuscation of a Turing machine  $M$  consists of two FHE ciphertexts  $g_1, g_2$ , each encrypting a description of  $M$  under a distinct public key, and an obfuscation of a certain (low-depth) verify-and-decrypt circuit. To evaluate an obfuscation of  $M$  on input  $x$ , a user will homomorphically evaluate the oblivious<sup>9</sup> universal Turing machine  $U(\cdot, x)$  on both ciphertexts  $g_1$  and  $g_2$ , and generate a  $\mathbf{P}$ -certificate  $\phi$  that the resulting ciphertexts  $e_1, e_2$  were computed correctly. Then, he will provide a low-depth proof  $\pi$  that the certificate  $\phi$  properly verifies (e.g., simply providing the entire circuit evaluation). The collection of  $e_1, e_2, \phi, \pi$  is then fed into the  $NC^1$ -obfuscated program, which checks the proofs, and if valid outputs the decryption of  $e_1$ .

Note that the use of computationally sound  $\mathbf{P}$ -certificates enables the size of the obfuscation of  $M$  to depend only on the *description size* of  $M$ , and not its runtime. This approach is not possible in the setting of *indistinguishability* obfuscation, as certificates of false statements *exist*, but are simply hard to find.

**Theorem 3.10.** *There exists a succinct extractability obfuscator  $e\mathcal{O}$  for bounded-input TM, as in Definition 3.8, assuming the existence of the following tools:*

- $e\mathcal{O}_{NC^1}$ : an extractability obfuscator for the class of circuits  $NC^1$ .
- $\mathcal{E} = (\text{Gen}_{\text{FHE}}, \text{Enc}_{\text{FHE}}, \text{Dec}_{\text{FHE}}, \text{Eval}_{\text{FHE}})$ : a fully homomorphic encryption scheme with decryption  $\text{Dec}$  in  $NC^1$ .
- $(\text{CRSGen}_{\text{cert}}, P_{\text{cert}}, V_{\text{cert}})$ : a  $\mathbf{P}$ -certificate system in the CRS model.

We first present the construction, and then analyze its properties below.

#### Extractability Obfuscator $e\mathcal{O}$ :

- $\text{Obfuscate}(1^k, M, s(k), t(k))$ : On input the security parameter  $1^k$ , description of a Turing machine  $M \in \mathcal{M}_k$ , and polynomial bounds on the size  $s = s(k)$  and runtime  $t = t(k)$  of Turing machines in the class  $\{\mathcal{M}_k\}$  to be obfuscated, do the following:
  1. Let  $U_k$  be the *oblivious* universal Turing machine (TM) that accepts as input a TM description  $T$  and a value  $v$ , and executes  $T$  on input  $v$  for  $t(k)$  steps.
  2. Sample two key pairs for the FHE scheme:  $(\text{pk}_{\text{FHE}}^1, \text{sk}_{\text{FHE}}^1) \leftarrow \text{Gen}_{\text{FHE}}(1^k)$  and  $(\text{pk}_{\text{FHE}}^2, \text{sk}_{\text{FHE}}^2) \leftarrow \text{Gen}_{\text{FHE}}(1^k)$ .
  3. Encrypt  $M$  under each FHE public key:  $g_1 \leftarrow \text{Enc}_{\text{FHE}}(\text{pk}_{\text{FHE}}^1, M)$  and  $g_2 \leftarrow \text{Enc}_{\text{FHE}}(\text{pk}_{\text{FHE}}^2, M)$ . Here we assume  $M$  is encoded in a canonical form as an  $\ell$ -bit string for use by the universal Turing machine  $U_k(\cdot, \cdot)$ .
  4. Sample a CRS for the  $\mathbf{P}$ -certificate system:  $\text{crs} \leftarrow \text{CRSGen}_{\text{cert}}(1^k)$ .

<sup>8</sup> $\mathbf{P}$ -certificates are immediately implied by any succinct non-interactive argument (SNARG) system for NP, but can additionally be based on *falsifiable* assumptions. We refer the reader to Section 2.3 for details.

<sup>9</sup>A Turing machine is said to be *oblivious* if the tape movements are independent of the input. Without obliviousness, one would be unable to homomorphically evaluate the Turing machine efficiently, as the location of the head of the Turing machine is encrypted.

**The program  $P1_{\text{sk}_{\text{FHE}}^1, g_1, g_2}$ :**

Given input  $(m, e_1, e_2, c, \phi, \pi)$ , the program proceeds as follows:

1. Check if  $\pi$  is a valid low-depth proof for the NP statement:

$$1 = V_{\text{cert}}(\text{crs}, c, (M_{\text{Eval}}, m, (e_1, e_2)), \phi),$$

where  $M_{\text{Eval}}$  is the computation that takes as input  $m$ , has  $\text{pk}_{\text{FHE}}^1, \text{pk}_{\text{FHE}}^2, g_1, g_2$  hardcoded, and homomorphically evaluates  $U_k(\cdot, m)$  on  $g_1, g_2$ .

2. If the check fails, output  $\perp$ ; otherwise, output  $\text{Dec}_{\text{FHE}}(\text{sk}_{\text{FHE}}^1, e_1)$ .

**The program  $P2_{\text{sk}_{\text{FHE}}^2, g_1, g_2}$ :**

Given input  $(m, e_1, e_2, c, \phi, \pi)$ , the program proceeds as follows:

1. Same as Step 1 of  $P1$ .
2. If the check fails, output  $\perp$ ; otherwise, output  $\text{Dec}_{\text{FHE}}(\text{sk}_{\text{FHE}}^2, e_2)$ .

**Figure 1:** The programs  $P1$  and  $P2$ .

5. Generate an  $NC^1$  obfuscation for the program  $P1_{\text{sk}_{\text{FHE}}^1, g_1, g_2}$  given in Figure 1, as

$$P \leftarrow e\mathcal{O}_{NC^1}(1^k, P1_{\text{sk}_{\text{FHE}}^1, g_1, g_2}).$$

(Note that  $P1$  also implicitly has hardcoded  $\text{crs}, \text{pk}_{\text{FHE}}^1, \text{pk}_{\text{FHE}}^2$ ).

6. Output the obfuscation  $\sigma = (P, \text{crs}, \text{pk}_{\text{FHE}}^1, \text{pk}_{\text{FHE}}^2, g_1, g_2)$ .
- Evaluate( $\sigma, m, s(k), t(k)$ ): On input an obfuscation  $\sigma = (P, \text{crs}, \text{pk}_{\text{FHE}}^1, \text{pk}_{\text{FHE}}^2, g_1, g_2)$ , program input  $m$ , and size and runtime bounds  $s = s(k), t = t(k)$  for the relevant class of Turing machines  $\{\mathcal{M}_k\}$ , do the following:

1. Homomorphically evaluate the  $m$ -evaluation function independently on the two FHE ciphertexts  $g_1, g_2$  (allegedly encrypting a Turing machine description): i.e.,

$$e_1 = \text{Eval}_{\text{FHE}}(\text{pk}_{\text{FHE}}^1, U_k(\cdot, m), g_1) \text{ and } e_2 = \text{Eval}_{\text{FHE}}(\text{pk}_{\text{FHE}}^2, U_k(\cdot, m), g_2),$$

where  $U_k$  is the oblivious universal Turing machine that accepts as input a TM description  $T$  and value  $v$ , and executes  $T$  on input  $v$  for  $t(k)$  steps. (See Remark 2.2 for discussion on homomorphic evaluation of an oblivious Turing machine of known runtime). Denote this computation (taking as input  $m$  and computing  $e_1, e_2$ ) by  $M_{\text{Eval}}$ . Note that  $M_{\text{Eval}}$  has  $\text{pk}_{\text{FHE}}^1, \text{pk}_{\text{FHE}}^2, g_1, g_2$  hardcoded.

2. Generate a  $\mathbf{P}$ -certificate that  $e_1$  and  $e_2$  were computed correctly. That is, taking a bound  $c \in \mathbb{N}$  for which  $M_{\text{Eval}}(m)$  outputs within  $|m|^c$  steps, let  $\phi \leftarrow P_{\text{cert}}(\text{crs}, c, (M_{\text{Eval}}, m, (e_1, e_2)))$ .
3. Compute a low-depth proof  $\pi$  that  $\phi$  is a valid  $\mathbf{P}$ -certificate: i.e.,  $V_{\text{cert}}(\text{crs}, c, (M_{\text{Eval}}, m, (e_1, e_2)), \phi) = 1$ . This can be achieved by simply providing the complete evaluation tableau of the  $V_{\text{cert}}$  circuit.
4. Run  $P(m, e_1, e_2, c, \phi, \pi)$ . (Recall  $P$  is an obfuscation of the program  $P1_{\text{sk}_{\text{FHE}}^1, g_1, g_2}$ , as described in Figure 1) and output the result.

*Proof of Theorem 3.10.* We now prove the correctness, succinctness, and security of  $e\mathcal{O}$ .

**Correctness.** First, note that the circuit evaluating the program  $P1_{\text{sk}_{\text{FHE}}^1, g_1, g_2}$  is in  $NC^1$ . Indeed, the program is composed of (1) verifying a low-depth proof, which is accomplished in  $NC^1$  by construction, and (2) evaluating the FHE decryption circuit, which is in  $NC^1$  by our choice of FHE scheme.

The correctness of the FHE encryption and evaluation algorithms implies that with overwhelming probability  $g_1$  is a valid encryption of the Turing machine description  $M$  and that  $e_1$  is an encryption of  $U_k(M, m) = M(m)$ . The correctness of the underlying  $NC^1$  obfuscator  $e\mathcal{O}_{NC^1}$  guarantees the obfuscation of the program  $P1_{\text{sk}_{\text{FHE}}^1, g_1, g_2}$  will evaluate the program correctly; the correctness of the  $\mathbf{P}$ -certificate scheme provides that the honestly generated certificate will properly verify; and the check step program itself will pass on honest execution. Therefore, the final output will be the desired evaluation  $M(m)$ .

**Succinctness.** We now analyze the description size and running time of the obfuscation  $M'$ .

Consider first the size of  $M'$ . Recall  $M'$  is composed of the following elements: a common reference string  $\text{crs}$  for the  $\mathbf{P}$ -certificate system, two FHE public keys  $\text{pk}^1, \text{pk}^2$ , two FHE ciphertexts  $g_1, g_2$  encrypting a description of  $M$ , and an obfuscation  $P$  of the program  $P1$  with respect to  $e\mathcal{O}_{NC^1}$ . We have that  $|\text{crs}|, |\text{pk}^1|, |\text{pk}^2| = \text{poly}(k)$ , and  $|g_1|, |g_2| = \text{poly}(k) \cdot s(k)$  (where  $s = s(k)$  is the maximum size of TMs in the class  $\mathcal{M}_k$ ). The underlying  $NC^1$  obfuscator has the property that  $|e\mathcal{O}_{NC^1}(C)| = \text{poly}(|C|)$  for any circuit  $C \in NC^1$ , so it remains to analyze the size of a circuit  $C$  representing the program  $P1$ .

The program  $P1$  proceeds in two steps. In the second step, it evaluates the decryption circuit of the FHE scheme on the ciphertext  $e_1$ . This decryption requires size  $\text{poly}(k) \cdot s(k)$ . In the first step of  $P1$ , it accepts and verifies the consistency of a (low-depth) proof  $\pi$  corresponding to the computation of  $V_{\text{cert}}(\text{crs}, c, (\text{Eval}(U_k(\cdot, \cdot), g_1, g_2), m), (e_1, e_2)), \phi)$ , verifying that  $\phi$  is a valid  $\mathbf{P}$ -certificate for the correct homomorphic evaluation yielding  $e_1, e_2$ . The verification of  $\pi$  requires size equal to the size of  $V_{\text{cert}}$  on these inputs. By the efficient verification property of the  $\mathbf{P}$ -certificate system, this is bounded by  $g_V(k + |(\text{Eval}(U_k(\cdot, \cdot), g_1, g_2), m), (e_1, e_2))|)$ , where  $g_V$  is a fixed polynomial. This expression is dominated by the description size of the universal Turing machine  $U_k$  (which was chosen sufficiently large to evaluate any Turing machine  $M \in \mathcal{M}_k$ ). We have that  $|U_k| = \text{poly}(s(k))$ . Thus, putting these pieces together, it holds that the  $|M'| = \text{poly}(k, s(k))$ , as desired.

Consider now the runtime of  $M'$ . We analyze each step of  $\text{Evaluate}(\sigma, m)$ :

1. Homomorphic evaluation of  $U_k(\cdot, m)$  on  $g_1, g_2$ : By Remark 2.2, this takes place in time  $\text{poly}(k, t(k))$ , where  $t(k)$  was a bound on the maximum runtime of Turing machines  $M \in \mathcal{M}_k$ .
2. Generating a  $\mathbf{P}$ -certificate of correctness: By the properties of the  $\mathbf{P}$ -certificate system, this  $P_{\text{cert}}$  evaluation takes place in time bounded by  $g_P(k + |M_{\text{Eval}}| + T_{M_{\text{Eval}}}(m))$ , where  $g_P$  is a fixed polynomial (and  $M_{\text{Eval}}$  is the program corresponding to the homomorphic evaluation in Step 1). From above, we have that this is  $\text{poly}(k, t(k))$ .
3. Compute a low-depth proof that the  $\mathbf{P}$ -certificate correctly verifies: This corresponds precisely to the time of evaluating  $V_{\text{cert}}$ . By the efficient verification property of the  $\mathbf{P}$ -certificate system, this takes place in time  $g_V(k + |(M_{\text{Eval}}, m), (e_1, e_2)|)$ , where  $g_V$  is a fixed polynomial. Since  $|e_1|, |e_2| = \text{poly}(k) \cdot s(k)$  and  $|M_{\text{Eval}}| = \text{poly}(s(k))$ , it follows that this step takes time  $\text{poly}(s(k))$ .
4. Run the obfuscated program  $P(m, e_1, e_2, c, \phi, \pi)$ : By assumed properties of the underlying  $NC^1$  extractability obfuscator  $e\mathcal{O}_{NC^1}$ , the runtime of  $P$  is polynomial in the runtime of the original program  $P1$ . From the analysis above (since the size of a circuit corresponds also to its runtime), the runtime of  $P1$  is thus  $\text{poly}(s(k))$ .



Combining the runtime of each of the above steps, we conclude that the runtime of the obfuscated program  $M'$  is  $\text{poly}(k, s(k), t(k))$ .

**Security.** Fix any PPT adversary  $\mathcal{A}$  and polynomial  $p(k)$ . In order to construct the desired extractor  $E$ , for any pair of Turing machines  $M_0, M_1 \in \mathcal{M}_k$  we consider a sequence of hybrid experiments, gradually modifying an obfuscation of  $M_0$  to  $M_1$ . From these intermediate experiments, we will derive a collection of different extraction strategies; the ultimate extractor  $E$  will randomly select one of these strategies in order to attempt extraction. We will show that if  $\mathcal{A}$  can distinguish an obfuscation of  $M_0$  from one of  $M_1$ , then it must successfully distinguish between some adjacent pair of hybrid experiments, which will imply the corresponding “sub”-extractor strategy will succeed. Since  $E$  chooses among these sub-strategies at random, it will choose the successful strategy with probability at least  $1/(\# \text{ hybrids})$ .

Explicitly, consider the following sequence of hybrids:

- **Hyb<sub>0</sub>**: This hybrid corresponds to an honest execution of  $e\mathcal{O}$  to obfuscate  $M_0$ .
- **Hyb<sub>1</sub>**: Same as hybrid **Hyb<sub>0</sub>**, except that the second FHE ciphertext is now generated as an encryption of  $M_1$ :  $g_2 \leftarrow \text{Enc}_{\text{FHE}}(\text{pk}_{\text{FHE}}^2, M_1)$ .
- **Hyb<sub>2</sub>**: Same as hybrid **Hyb<sub>1</sub>**, except that the obfuscated program  $P$  is now generated as an obfuscation of  $P_2$  (which decrypts using the *second* FHE secret key). That is,  $P \leftarrow e\mathcal{O}_{\text{NC}^1}(P_2^{\text{sk}_{\text{FHE}}^2, g_1, g_2})$ .
- **Hyb<sub>3</sub>**: Same as hybrid **Hyb<sub>2</sub>**, except that the *first* FHE ciphertext is now also generated as an encryption of  $M_1$ :  $g_1 \leftarrow \text{Enc}_{\text{FHE}}(\text{pk}_{\text{FHE}}^1, M_1)$ .
- **Hyb<sub>4</sub>**: Same as hybrid **Hyb<sub>3</sub>**, except that the obfuscated program  $P$  is once again generated as an obfuscation of  $P_1$ : i.e.,  $P \leftarrow e\mathcal{O}_{\text{NC}^1}(P_1^{\text{sk}_{\text{FHE}}^1, g_1, g_2})$ . Note that this hybrid corresponds to an honest execution of  $e\mathcal{O}$  to obfuscate  $M_1$ .

**Step 1: Hyb<sub>0</sub> to Hyb<sub>1</sub>.** We argue that no PPT  $\mathcal{A}$  will be able to distinguish between these hybrids, and thus we need not address extraction in this case.

**Claim 3.11** (Security of FHE). *Assuming the FHE scheme  $\mathcal{E}$  is IND-CPA secure, then the outputs of **Hyb<sub>0</sub>** and **Hyb<sub>1</sub>** are computationally indistinguishable.*

*Proof.* Suppose there exists a PPT distinguisher  $\mathcal{A}$  between the two hybrids. We then demonstrate a PPT attacker  $\mathcal{A}_{\text{FHE}}$  who breaks the IND-CPA security of the FHE scheme.

$\mathcal{A}_{\text{FHE}}$  begins by executing  $\mathcal{A}$  and receiving a pair of Turing machine descriptions  $M_0, M_1 \in \mathcal{M}_k$ . In the FHE security game,  $\mathcal{A}_{\text{FHE}}$  receives a public key  $\text{pk}$  from the FHE challenger;  $\mathcal{A}_{\text{FHE}}$  submits  $M_0, M_1$  as his challenge message pair, and receives a ciphertext  $g'$  (corresponding to an encryption of one of the two messages).  $\mathcal{A}_{\text{FHE}}$  sets  $\text{pk}_{\text{FHE}}^2 = \text{pk}$  and  $g_2 = g'$ . It further samples a fresh FHE key pair  $(\text{pk}_{\text{FHE}}^1, \text{sk}_{\text{FHE}}^1) \leftarrow \text{Gen}_{\text{FHE}}(1^k)$ , generates an encryption of  $M_0$  under this public key, as  $g_1 \leftarrow \text{Enc}_{\text{FHE}}(\text{pk}_{\text{FHE}}^1, M_0)$ , and generates an obfuscation of the circuit  $P_1^{\text{sk}_{\text{FHE}}^1, g_1, g_2}$  as  $P \leftarrow e\mathcal{O}_{\text{NC}^1}(P_1^{\text{sk}_{\text{FHE}}^1, g_1, g_2})$ . Note that this requires only knowledge of the *first* FHE secret key (and not the second, which is unknown to  $\mathcal{A}_{\text{FHE}}$ ).  $\mathcal{A}_{\text{FHE}}$  then submits the tuple  $(P, \text{pk}_{\text{FHE}}^1, \text{pk}_{\text{FHE}}^2, g_1, g_2)$  to the adversary  $\mathcal{A}$ .

Note that if  $g'$  was an encryption of  $M_0$ , then this tuple is distributed exactly as the output of hybrid **Hyb<sub>0</sub>**, whereas if  $g'$  was an encryption of  $M_1$  the tuple is distributed exactly as in hybrid **Hyb<sub>1</sub>**. Thus, the advantage of  $\mathcal{A}_{\text{FHE}}$  in the FHE security game is identical to the distinguishing advantage of  $\mathcal{A}$ , as desired.  $\square$

**Step 2: Hyb<sub>1</sub> to Hyb<sub>2</sub>.** We show that any adversary who distinguishes between these hybrids necessarily implies an extractor who succeeds with noticeable probability in extracting an input  $x$  on which  $M_0$  and  $M_1$  disagree.

**Claim 3.12** (Security of  $e\mathcal{O}_{NC^1}$ , soundness of  $\mathbf{P}$ -certificates). *Suppose  $e\mathcal{O}_{NC^1}$  is an extractability obfuscator for the circuit class  $NC^1$ , and  $(P_{\text{cert}}, V_{\text{cert}})$  is a sound  $\mathbf{P}$ -certificate system. Then for any PPT adversary  $\mathcal{A}$  and polynomial  $p_1(k)$  there exists a PPT extractor  $E_{\text{Hyb}_1}$  and polynomial  $q_1(k)$  such that, for every pair  $M_0, M_1 \in \mathcal{M}_k$  and every auxiliary input  $z$ ,*

$$\begin{aligned} \Pr \left[ b \leftarrow \{1, 2\}; H \leftarrow \text{Hyb}_b(M_0, M_1) : \mathcal{A}(1^k, H, M_0, M_1, z) = b \right] &\geq \frac{1}{2} + \frac{1}{p_1(k)} \\ \implies \Pr[x \leftarrow E_{\text{Hyb}_1}(1^k, M_0, M_1, z) : M_0(x) \neq M_1(x)] &\geq \frac{1}{q_1(k)}. \end{aligned}$$

*Proof.* Fix a PPT adversary  $\mathcal{A}$  in the Hyb<sub>1</sub> versus Hyb<sub>2</sub> distinguishing game. For every choice of values  $z_v := (\text{pk}_{\text{FHE}}^1, \text{pk}_{\text{FHE}}^2, g_1, g_2)$ , consider the following adversary  $\mathcal{A}_{NC^1}$  (induced by  $\mathcal{A}$ ) on the underlying  $NC^1$  extractability obfuscator  $e\mathcal{O}_{NC^1}$  for the pair of  $NC^1$  programs  $P1_{\text{sk}_{\text{FHE}}^1, g_1, g_2}$  and  $P2_{\text{sk}_{\text{FHE}}^2, g_1, g_2}$ :

The  $e\mathcal{O}_{NC^1}$  adversary  $\mathcal{A}_{NC^1}(1^k, P1_{\text{sk}_{\text{FHE}}^1, g_1, g_2}, P2_{\text{sk}_{\text{FHE}}^2, g_1, g_2}, (z_{\mathcal{A}}, z_v))$ :

1.  $\mathcal{A}_{NC^1}$  receives a challenge obfuscation  $P$ , generated as either  $P \leftarrow e\mathcal{O}_{NC^1}(P1_{\text{sk}_{\text{FHE}}^1, g_1, g_2})$  or  $P \leftarrow e\mathcal{O}_{NC^1}(P2_{\text{sk}_{\text{FHE}}^2, g_1, g_2})$ .
2.  $\mathcal{A}_{NC^1}$  runs the adversary  $\mathcal{A}$  on input  $(1^k, H, M_0, M_1, z_{\mathcal{A}})$ , where  $H$  is the tuple of values  $H = (P, \text{pk}_{\text{FHE}}^1, \text{pk}_{\text{FHE}}^2, g_1, g_2)$ , where  $\text{pk}_{\text{FHE}}^1, \text{pk}_{\text{FHE}}^2, g_1, g_2$  are given in  $z_v$  (Note that for appropriate distribution of  $z_v$ , this tuple will correspond to the output distribution of either Hyb<sub>1</sub> or Hyb<sub>2</sub>). Denote the output of  $\mathcal{A}$  by  $\text{guess} \in \{1, 2\}$ .
3.  $\mathcal{A}_{NC^1}$  outputs  $\text{guess}$  as his own prediction in the  $e\mathcal{O}_{NC^1}$  distinguishing game.

Note that if the values in  $z_v$  are generated in the following way, then the tuple of values  $\sigma$  given to  $\mathcal{A}$  in Step 2 is distributed identically to the output of either Hyb<sub>1</sub> or Hyb<sub>2</sub>, depending exactly on whether the challenge obfuscation  $P$  was generated as an obfuscation of  $P1$  or  $P2$ .

Sampling procedure  $\text{SampleZ}(1^k)$  for  $z_v$ :

1. Sample two key pairs for the FHE scheme:  $(\text{pk}_{\text{FHE}}^1, \text{sk}_{\text{FHE}}^1) \leftarrow \text{Gen}_{\text{FHE}}(1^k)$  and  $(\text{pk}_{\text{FHE}}^2, \text{sk}_{\text{FHE}}^2) \leftarrow \text{Gen}_{\text{FHE}}(1^k)$ .
2. Encrypt a description of  $M_0$  under the first FHE public key:  $g_1 \leftarrow \text{Enc}_{\text{FHE}}(\text{pk}_{\text{FHE}}^1, M_0)$ , and encrypt a description of  $M_1$  under the second FHE key:  $g_2 \leftarrow \text{Enc}_{\text{FHE}}(\text{pk}_{\text{FHE}}^2, M_1)$ .
3. Output  $z_v = (\text{pk}_{\text{FHE}}^1, \text{pk}_{\text{FHE}}^2, g_1, g_2)$ .

Thus, if  $\mathcal{A}(1^k, H, M_0, M_1, z_{\mathcal{A}})$  distinguishes between  $H \leftarrow \text{Hyb}_1(M_0, M_1)$  and  $H \leftarrow \text{Hyb}_2(M_0, M_1)$  with advantage  $1/p(k)$  over the entire randomness of the experiment, then with probability at least  $1/2p(k)$  over the choice of  $z_v \leftarrow \text{SampleZ}(1^k)$ , it must hold that  $\mathcal{A}$  distinguishes between such  $H$  conditioned on the value  $z_v$ , with advantage  $1/2p(k)$ .

Therefore, by the security of the  $NC^1$  obfuscator  $e\mathcal{O}_{NC^1}$ , there exists a PPT extractor  $E_{NC^1}$  and polynomial  $q'(k)$  such that, with probability  $1/2p(k)$  over the choice of  $z_v \leftarrow \text{SampleZ}(1^k)$ ,  $E_{NC^1}$  extracts a value  $x \leftarrow E_{NC^1}(1^k, P1_{\text{sk}_{\text{FHE}}^1, g_1, g_2}, P2_{\text{sk}_{\text{FHE}}^2, g_1, g_2}, z_v)$  for which the circuits disagree (i.e.,  $P1_{\text{sk}_{\text{FHE}}^1, g_1, g_2}(x) \neq P2_{\text{sk}_{\text{FHE}}^2, g_1, g_2}(x)$ ) with success probability  $1/q'(k)$ .

It remains to show that any such input  $x$  must contain a value  $m$  for which  $M_0(m) \neq M_1(m)$ . Indeed, if this is the case, then the algorithm  $E_{\text{Hyb}_1}$  which first samples  $z_v \leftarrow \text{SampleZ}$  and then executes  $E_{NC^1}$  on the appropriate set of values will have successfully extracted with overall noticeable success probability  $\frac{1}{p(k)} \frac{1}{q'(k)}$ .

Recall that the input of  $P1$  (or  $P2$ ) is a tuple  $(m, e_1, e_2, c, \phi, \pi)$  where  $m$  is an input to a circuit in  $\mathcal{C}_k$ ,  $e_1, e_2$  are allegedly homomorphically evaluated ciphertexts,  $c \in \mathbb{N}$  is a runtime bound on this evaluation procedure,  $\phi$  is a  $\mathbf{P}$ -certificate that  $e_1, e_2$  were computed correctly, and  $\pi$  is a low-depth proof that  $\phi$  properly verifies.

First, note that Step 1 of the programs  $P1_{\text{sk}_{\text{FHE}}^1, g_1, g_2}$  and  $P2_{\text{sk}_{\text{FHE}}^2, g_1, g_2}$  are identical, since they each have the same pair of ciphertexts  $g_1, g_2$  hardcoded. So for any input tuple  $(m, e_1, e_2, c, \phi, \pi)$ , this tuple will either pass the verification in both  $P1$  and  $P2$  or fail in both. Thus, in order for  $P1_{\text{sk}_{\text{FHE}}^1, g_1, g_2}(x) \neq P2_{\text{sk}_{\text{FHE}}^2, g_1, g_2}(x)$  for some  $x = (m, e_1, e_2, c, \phi, \pi)$ , it must be that (1) the low-depth proof  $\pi$  verifies correctly (otherwise both programs output  $\perp$ ), and (2)  $\text{Dec}_{\text{FHE}}(\text{sk}_{\text{FHE}}^1, e_1) \neq \text{Dec}_{\text{FHE}}(\text{sk}_{\text{FHE}}^2, e_2)$ .

Since the proof  $\pi$  verifies correctly, it must be that indeed  $1 = V_{\text{cert}}(\text{crs}, c, (M_{\text{Eval}}, m, (e_1, e_2)), \phi)$ . That is,  $\phi$  must be a valid  $\mathbf{P}$ -certificate that  $e_1, e_2$  were generated as the result of homomorphically evaluating  $U_k(\cdot, m)$  on the (hardcoded) FHE ciphertexts  $g_1, g_2$ , respectively (recall this is the definition of  $M_{\text{Eval}}$ ). By the soundness of the  $\mathbf{P}$ -certificate system, since  $E_{NC^1}$  is an efficient PPT algorithm, this means that with all but negligible probability this is truly the case: i.e.,  $e_1 = \text{Eval}_{\text{FHE}}(\text{pk}_{\text{FHE}}^1, U_k(\cdot, m), g_1)$  and  $e_2 = \text{Eval}_{\text{FHE}}(\text{pk}_{\text{FHE}}^2, U_k(\cdot, m), g_2)$ . Recall that  $g_1$  and  $g_2$  were generated as encryptions of  $M_0$  and  $M_1$ . By the correctness of the FHE decryption and evaluation algorithms, this means with all but negligible probability that  $\text{Dec}_{\text{FHE}}(\text{sk}_{\text{FHE}}^1, e_1) = M_0(m)$  and  $\text{Dec}_{\text{FHE}}(\text{sk}_{\text{FHE}}^2, e_2) = M_1(m)$ . Thus, the value  $m$  satisfies  $M_0(m) \neq M_1(m)$ , as desired.  $\square$

**Step 3: Hyb<sub>2</sub> to Hyb<sub>3</sub>.** As in Step 1, we argue that no PPT  $\mathcal{A}$  will be able to distinguish between these hybrids, and thus we need not address extraction in this case.

**Claim 3.13** (Security of FHE). *Assuming the FHE scheme  $\mathcal{E}$  is IND-CPA secure, then the outputs of Hyb<sub>2</sub> and Hyb<sub>3</sub> are computationally indistinguishable.*

*Proof.* The claim follows in a nearly identical fashion to that of Claim 3.11.  $\square$

**Step 4: Hyb<sub>3</sub> to Hyb<sub>4</sub>.** This step directly mirrors Step 2.

**Claim 3.14** (Security of  $e\mathcal{O}_{NC^1}$ , soundness of  $\mathbf{P}$ -certificates). *Suppose that  $e\mathcal{O}_{NC^1}$  is an extractability obfuscator for the circuit class  $NC^1$ , and  $(P_{\text{cert}}, V_{\text{cert}})$  is a sound  $\mathbf{P}$ -certificate system. Then for any PPT adversary  $\mathcal{A}$  there exists a PPT extractor  $E_{\text{Hyb}_3}$  for which the following holds: For every polynomial  $p(k)$ , there exists a polynomial  $q'(k)$  such that for every pair  $M_0, M_1 \in \mathcal{M}_k$  and every auxiliary input  $z$ ,*

$$\Pr \left[ b \leftarrow \{3, 4\}; H \leftarrow \text{Hyb}_b(M_0, M_1) : \mathcal{A}(1^k, H, z) = b \right] \geq \frac{1}{2} + \frac{1}{p(k)}$$

$$\Pr[x \leftarrow E_{\text{Hyb}_3}(1^k, M_0, M_1, z) : M_0(x) \neq M_1(x)] \geq \frac{1}{q'(k)}.$$

*Proof.* The claim follows in an identical fashion to that of Claim 3.12.  $\square$

Our final extractor  $E$  works by choosing at random  $a \leftarrow \{1, 3\}$  and then executing the sub-extractor strategy  $E_{\text{Hyb}_a}$ .

The proof of Theorem 3.10 follows from Claims 3.11-3.14, in the following fashion. For any polynomial  $p_{\mathcal{A}}(k)$ , take  $p_E(k)$  to be the polynomial  $\frac{1}{2}q'(k)$  given by the  $e\mathcal{O}_{NC^1}$  extraction security for distinguishing success advantage  $\frac{1}{4}p_{\mathcal{A}}(k)$  (i.e., the extraction probability of the  $e\mathcal{O}_{NC^1}$  extractor is  $q'(k)$  if the  $e\mathcal{O}_{NC^1}$  adversary distinguishes with advantage  $\frac{1}{4}p_{\mathcal{A}}(k)$ ).

Suppose there exists a pair of Turing machines  $M_0, M_1 \in \mathcal{C}_k$  and auxiliary input  $z$  such that  $\mathcal{A}$  distinguishes between  $e\mathcal{O}$ -obfuscations of  $M_0$  and  $M_1$  with advantage  $p_{\mathcal{A}}(k)$ :

$$\Pr \left[ b \leftarrow \{0, 1\}; C' \leftarrow e\mathcal{O}(1^k, C_b) : \mathcal{A}(1^k, C', z) = b \right] \geq \frac{1}{2} + \frac{1}{p_{\mathcal{A}}(k)}.$$

Then for some  $i \in \{0, 1, 2, 3\}$ ,  $\mathcal{A}$  must successfully distinguish between adjacent hybrids  $\text{Hyb}_i$  and  $\text{Hyb}_{i+1}$  with advantage  $\frac{1}{4}p_{\mathcal{A}}(k)$ . By Claims 3.11 and 3.13, if it is the case that  $i = 0$  or  $2$  then we have a contradiction. If  $i = 1$ , then by Claim 3.12 the extraction strategy  $E_{\text{Hyb}_1}$  will succeed with probability  $q'(k)$ . Finally, if  $i = 3$ , then by Claim 3.14 the extraction strategy  $E_{\text{Hyb}_3}$  will succeed with probability  $q'(k)$ .

Therefore, with probability  $\frac{1}{2}$  the constructed extractor  $E$  will correctly guess the correct value of  $i \in \{1, 3\}$ , in which case it will succeed in extraction with probability  $q'(k)$ , hence yielding an overall extraction probability of  $\frac{1}{2}q'(k)$ , as desired.  $\square$

We remark that by replacing the  $\mathbf{P}$ -certificates with succinct non-interactive arguments of knowledge (SNARKs), then the resulting extractability obfuscator is secure for all polynomial-size Turing machines of possibly unbounded input size.

**Corollary 3.15.** *Based on any extractability obfuscator for the class of circuits  $NC^1$ , fully homomorphic encryption, and succinct non-interactive arguments of knowledge (SNARKs), there exists an extractability obfuscator for TM, as in Definition 3.7.*

We also observe that by using a *leveled* FHE, and removing the  $\mathbf{P}$ -certificates from the construction, we can still achieve extractability obfuscation for  $P/\text{poly}$ .

**Remark 3.16** ((Non-succinct) Extractability Obfuscation from Weaker Assumptions). One may remove the assumption of  $\mathbf{P}$ -certificates, and remove the circular security assumption of the FHE scheme (yielding only *leveled* FHE), in exchange for losing succinctness of the resulting obfuscator.

More explicitly, the above extractability obfuscator construction  $e\mathcal{O}$  can be modified as follows. Instead of generating a  $\mathbf{P}$ -certificate that the homomorphic evaluation of  $U_k$  was performed correctly and then computing a low-depth proof that the resulting  $\mathbf{P}$ -certificate properly verifies, simply generate a (large) low-depth proof of correctness of the homomorphic evaluation directly. Further, in the place of FHE, simply sample and utilize keys for a leveled FHE scheme with sufficient levels to support homomorphic evaluation of  $U_k$ . The resulting transformation  $e\mathcal{O}'$  still satisfies the required correctness and security properties, but no longer achieves succinctness (i.e., the size of the obfuscated Turing machine depends polynomially on its runtime):

- The leveled FHE scheme will now require larger public key sizes, proportional to the *runtime* of the Turing machine  $M$  to be obfuscated. Recall the obfuscated program  $e\mathcal{O}'(1^k, M)$  contains within it two such (leveled) FHE public keys.
- By removing  $\mathbf{P}$ -certificates, the obfuscation  $M'$  will now contain an underlying  $NC^1$  obfuscation  $P$  of a program that verifies the entire homomorphic evaluation of  $U_k$  gate by gate. While

this computation is low depth (indeed, the circuit still lies in  $NC^1$ ), the *size* of the circuit, and hence the size of  $P$ , will grow at least linearly in the *runtime* of  $U_k$  instead of just its size.

**Theorem 3.17.** *Based on any extractability obfuscator for the class of circuits  $NC^1$ , and leveled fully homomorphic encryption, there exists a (non-succinct) extractability obfuscator for  $P/\text{poly}$ .*

## 4 Functional Witness Encryption

We put forth the notion of *functional witness encryption (FWE)*. An FWE scheme enables one to encrypt a message  $m$  with respect to an  $NP$  language  $L$ , instance  $x$  and a function  $f$ , such that anyone that has, and *only* those that have, a witness  $w$  for  $x \in L$  can recover  $f(m, w)$ . More precisely, our security definition requires that if you can distinguish encryptions of two messages  $m_0, m_1$ , then you must know a witness  $w$  for  $x \in L$  such that  $f(m_0, w) \neq f(m_1, w)$ .

For example, an FWE scheme would allow one to encrypt the nodes of a large graph in such a way that anybody (and *only those*) who knows a clique in the graph can decrypt the labels *on the corresponding clique*.

**Definition 4.1** (Functional Witness Encryption). A *functional witness encryption* scheme for an  $NP$  language  $L$  (with corresponding witness relation  $R$ ) and class of Turing machines  $\{\mathcal{M}_k\}_{k \in \mathbb{N}}$ , consists of the following two polynomial-time algorithms:

- $\text{Enc}(1^k, x, m, M)$ : The encryption algorithm takes as input the security parameter  $1^k$ , an unbounded-length string  $x$ , a message  $m \in \text{MSG}$  for some message space  $\text{MSG}$ , and a Turing machine description  $M \in \mathcal{M}_k$ , and outputs a ciphertext  $c$ .
- $\text{Dec}(c, w)$ : The decryption algorithm takes as input a ciphertext  $c$  and an unbounded-length string  $w$ , and outputs an evaluation  $m'$  or the symbol  $\perp$ .

satisfying the following conditions:

**Correctness:** There exists a negligible function  $\text{negl}(k)$  such that for every security parameter  $k$ , for any message  $m \in \text{MSG}$ , for any Turing machine  $M \in \mathcal{M}_k$ , and for any  $x \in L$  such that  $R(x, w)$  holds, we have that

$$\Pr \left[ \text{Dec}(\text{Enc}(1^k, x, m, M), w) = M(m, w) \right] = 1 - \text{negl}(k).$$

**Security:** For every PPT adversary  $\mathcal{A}$  and polynomials  $p(k), \ell(k)$ , there exists a PPT extractor  $E$  and polynomial  $q(k)$  such that for every security parameter  $k$ , every pair of messages  $m_0, m_1 \in \text{MSG}_k$ , every Turing machine  $M \in \mathcal{M}_k$ , string  $x$ , and auxiliary input  $z$  of length at most  $\ell(k)$ ,

$$\begin{aligned} & \Pr \left[ b \leftarrow \{0, 1\}; c \leftarrow \text{Enc}(1^k, x, m_b, M) : \mathcal{A}(1^k, c, z) = b \right] \geq \frac{1}{2} + \frac{1}{p(k)} \\ \implies & \Pr \left[ w \leftarrow E(1^k, p(k), x, m_0, m_1, M, z) : M(m_0, w) \neq M(m_1, w) \right] \geq \frac{1}{q(k)}. \end{aligned}$$

**Definition 4.2** ((Succinct) FWE for  $NP$  and  $P/\text{poly}$ ). PPT algorithms  $(\text{Enc}_{NP}^{P/\text{poly}}, \text{Dec}_{NP}^{P/\text{poly}})$  are said to compose a *functional witness encryption scheme for  $NP$  and  $P/\text{poly}$*  if the following holds:

For every  $NP$  relation  $R$ , and every class of Turing machines  $\{\mathcal{M}_k\}$  with maximum description size  $s(k)$  and maximum runtime  $t(k)$  polynomial in  $k$ , the pair of algorithms induced by

$$\begin{aligned} \text{Enc}_{R,\{\mathcal{M}_k\}}(\cdot, \cdot, \cdot, \cdot) &:= \text{Enc}_{NP}^{P/\text{poly}}\left((\cdot, \cdot, \cdot, \cdot), R, s(k), t(k)\right), \\ \text{Dec}_{R,\{\mathcal{M}_k\}}(\cdot, \cdot) &:= \text{Dec}_{NP}^{P/\text{poly}}\left((\cdot, \cdot), R, s(k), t(k)\right) \end{aligned}$$

is a secure FWE scheme for language  $R$  and class of Turing machines  $\{\mathcal{M}_k\}$ .

$(\text{Enc}_{NP}^{P/\text{poly}}, \text{Dec}_{NP}^{P/\text{poly}})$  is further said to be *succinct* if there exist polynomials  $p_s$  and  $p_t$  such that for every  $NP$  relation  $R$ , every class of Turing machines  $\{\mathcal{M}_k\}$  with polynomial size and runtime  $s(k), t(k)$ , every  $k \in \mathbb{N}$ , every polynomial-size string  $x$ , every  $m \in \text{MSG}$ , and every Turing machine  $M \in \mathcal{M}_k$ , the encryption runtime

$$c \leftarrow \text{Enc}_{NP}^{P/\text{poly}}\left((1^k, x, m, M), R, s(k), t(k)\right)$$

is bounded by  $p_t(k, |R|, s(k), t(k))$ , and the corresponding ciphertext size satisfies

$$|c| \leq p_s(k, |R|, s(k)).$$

In particular, the ciphertext size depends only on the *description size* of the supported Turing machines, without turning size into runtime.

We demonstrate that FWE is, in fact, *equivalent* to extractability obfuscation, up to a simple transformation.

**Theorem 4.3** (Equivalence of FWE and Extractability Obfuscation). *The existence of the following two primitives is equivalent:*

1. *Succinct functional witness encryption for  $NP$  and  $P/\text{poly}$ , as in Definition 4.2.*
2. *Succinct extractability obfuscation for  $P/\text{poly}$ , as in Definition ??.*

Roughly, given an extractability obfuscator  $e\mathcal{O}$ , an FWE encryption of the message  $m$ , for the language  $L$ , instance  $x$  and function  $f$  will be the obfuscation of the program that on input  $w$  outputs  $f(m, w)$  if  $w$  is a valid witness for  $x \in L$ . On the other hand, given a general-purpose FWE, to obfuscate a program  $\Pi$ , let  $f$  be the universal circuit that on input  $(\Pi, y)$  runs  $\Pi$  on input  $y$ , let  $L$  be the trivial language where every witness is valid, and output a FWE of the message  $\Pi$ .

*Proof of Theorem 4.3.* We prove equivalence via two implications.

1. (FWE  $\Rightarrow$  Ext-Obf):

Suppose there exists a succinct FWE scheme  $(\text{Enc}_{NP}^{P/\text{poly}}, \text{Dec}_{NP}^{P/\text{poly}})$  for  $NP$  and  $P/\text{poly}$ . We construct the desired extractability obfuscator  $e\mathcal{O} = (\text{Obfuscate}, \text{Evaluate})$ .

- **Obfuscate.** Given size and runtime bounds  $s(k), t(k)$  for a class of Turing machines  $\{\mathcal{M}_k\}$  with respect we wish to obfuscate, take  $U_k^{s,t}$  to be the universal Turing machine accepting input TMs of size  $s(k)$  and evaluating them for  $t(k)$  steps; denote the size and runtime of  $U_k^{s,t}$  by  $s' = s'(k), t' = t'(k)$ . Define  $\text{Obfuscate}_{\{\mathcal{M}_k\}}$  as follows:

$\text{Obfuscate}_{\{\mathcal{M}_k\}}(1^k, M)$ : On input the security parameter  $1^k$  and Turing machine description  $M$ , generate an FWE encryption of the message  $M$  (i.e., the Turing machine description), with respect to the trivial  $NP$  relation  $R_1(x, w) = 1 \forall x, w$ , an arbitrary statement  $x$  (say  $x = 0$ ) and the function  $U_k^{s,t}$ . That is,

$$c \leftarrow \text{Enc}_{R_1, \{\mathcal{M}'_k\}}(1^k, 0, M, U_k),$$

where  $\{\mathcal{M}'_k\}$  denotes the class of Turing machines of size and runtime bounded by  $s'(k), t'(k)$ . Output  $c$  as the obfuscation of  $M$ .

- **Evaluate.** Given size and runtime bounds  $s(k), t(k)$  for a class of Turing machines  $\{\mathcal{M}_k\}$  with respect to which we wish to obfuscate, define  $\text{Evaluate}_{\{\mathcal{M}_k\}}$  as follows:

$\text{Evaluate}_{\{\mathcal{M}_k\}}(c, w)$ : On input an obfuscation  $c$ , evaluation input  $w$ , and size and runtime bounds  $s = s(k), t = t(k)$  for the class of TMs with respect to which we wish to obfuscate, do the following. Run the decryption algorithm on  $c$  using witness  $w$ . That is, output

$$y = \text{Dec}_{R_1, \{\mathcal{M}'_k\}}(c, w)$$

as the evaluation of the obfuscated program on input  $w$ .

*Correctness of  $e\mathcal{O}$* : By the correctness of the FWE scheme, since  $w$  is (trivially) a valid witness for the statement  $x = 0$  under relation  $R_1$ , and since the program  $U_k^{s,t}$  has size and runtime bounded by  $s'(k), t'(k)$  (and thus  $U_k \in \{\mathcal{M}'_k\}$ ), it holds with overwhelming probability that

$$\text{Dec}_{R_1, \{\mathcal{M}'_k\}}\left(\text{Enc}_{R_1, \{\mathcal{M}'_k\}}(1^k, 0, M, U_k), w\right) = U_k(M, w),$$

which is precisely the desired evaluation  $M(w)$ .

*Succinctness of  $e\mathcal{O}$* : Let  $p_s, p_t$  be the polynomials dictating the ciphertext size and encryption runtime of the succinct FWE scheme, as in Definition 4.2. In particular, for  $NP$  relation  $R_1$  and class of Turing machines  $\{\mathcal{M}'_k\}$  defined above (namely, containing Turing machines of size and runtime  $s'(k), t'(k)$  such that  $\mathcal{M}'_k$  contains the universal Turing machine  $U_k$  of sufficient size to evaluate all TMs  $M \in \mathcal{M}_k$ ), it holds that  $\text{Enc}_{R_1, \{\mathcal{M}'_k\}}$  runs in time  $p_t(k, |R_1|, s'(k), t'(k))$  and outputs ciphertexts of size  $p_s(k, |R_1|, s'(k))$ . Note that these correspond directly to the runtime and output size of the constructed obfuscator  $e\mathcal{O}$ .

Now, the trivial relation  $R_1$  can be represented in constant size. And, the desired universal Turing machine can always be chosen with  $s'(k) \in \tilde{O}(s(k))$  and  $t'(k) \in \tilde{O}(t(k))$ . Thus, for any choice of fixed polynomial  $q(k) \in k^{\omega(1)}$ , if we define the polynomials  $p'_t(k) := p_t(q(k))$  and  $p'_s(k) := p_s(q(k))$ , then it follows that the runtime and output size of the constructed obfuscator  $e\mathcal{O}$  are bounded by  $p'_t(k, s(k), t(k))$  and  $p'_s(k, s(k))$ , as required.

*Security of  $e\mathcal{O}$* : Fix any PPT extractability obfuscation adversary  $\mathcal{A}$  and polynomial  $p(k)$ . By the construction of  $e\mathcal{O}$ , it is the case that  $\mathcal{A}$  is also a valid adversary in the FWE security game. Thus, by the (extractability) security of the FWE scheme, there exists a PPT extractor  $E$  and polynomial  $q'(k)$  such that for every pair of messages  $M_0, M_1$ , every Turing machine  $T \in \mathcal{M}'_k$  (in particular, for  $U_k \in \mathcal{M}'_k$ ), every statement  $x$  (in particular, for  $x = 0$ ), and all auxiliary input  $z$ , if  $\mathcal{A}$  can distinguish between FWE ciphertexts  $\text{Enc}(1^k, 0, M_0, U_k)$  and  $\text{Enc}(1^k, 0, M_1, U_k)$  with advantage  $1/q(k)$ , then  $E$  successfully extracts a witness  $w$  for which  $U_k(M_0, w) \neq U_k(M_1, w)$ . But, these ciphertexts correspond directly to the distribution of obfuscations  $e\mathcal{O}(M_0)$  and  $e\mathcal{O}(M_1)$ . Further,  $U_k(M_0, w) = M_0(w)$  and  $U_k(M_1, w) = M_1(w)$ . Thus, it follows that if  $\mathcal{A}$  distinguishes between  $e\mathcal{O}(M_0)$  and  $e\mathcal{O}(M_1)$  with advantage  $1/q(k)$  then  $E$  extracts an input  $w$  for which  $M_0(w) \neq M_1(w)$  with probability  $1/q'(k)$ , as desired.

## 2. (Ext-Obf $\Rightarrow$ FWE):

Suppose there exists a succinct extractability obfuscator  $e\mathcal{O}^{P/\text{poly}}$  for  $P/\text{poly}$ . We construct the desired FWE scheme (Enc, Dec).

- **Encrypt.** Given an  $NP$  relation  $R$  and size and runtime bounds  $s(k), t(k)$  on the desired class of supported Turing machine computations  $\{\mathcal{M}_k\}$ , define  $\text{Enc}_{R, \{\mathcal{M}_k\}}$  as follows:

$\text{Enc}_{R,\{\mathcal{M}_k\}}(1^k, x, m, M)$ : On input the security parameter  $1^k$ , an unbounded-length string  $x$ , message  $m \in \text{MSG}$ , and Turing machine  $M \in \mathcal{M}_k$ , define  $\{\mathcal{M}'_k\}$  to be the class of Turing machines whose size and runtime are bounded by  $(s(k) + |R|)$  and  $(t(k) + |R|)$ , and generate an obfuscation

$$\sigma \leftarrow \text{Obfuscate}_{\{\mathcal{M}'_k\}}(1^k, P_{R,x,m,M})$$

of the following program  $P_{R,x,m,M}(w)$ :

- (a) If  $R(x, w) \neq 1$ , then output  $\perp$ .
- (b) Else, evaluate and output  $M(m, w)$ .

Output the obfuscation  $\sigma$  as the desired ciphertext.

- **Decrypt.** Given  $NP$  relation  $R$  and size and runtime bounds  $s(k), t(k)$  on the desired class of supported Turing machine computations  $\{\mathcal{M}_k\}$ , define  $\text{Dec}_{R,\{\mathcal{M}_k\}}$  as follows:

$\text{Dec}_{R,\{\mathcal{M}_k\}}(\sigma, w)$ : On input a ciphertext  $\sigma$  and witness  $w$ , execute the obfuscated program  $\sigma$  on input  $w$ : i.e., output  $y = \text{Evaluate}_{\{\mathcal{M}'_k\}}(\sigma, w)$ .

*Correctness of (Enc, Dec)*: Note that  $P_{R,x,m,M} \in \mathcal{M}'_k$ , since its size and runtime are given by  $(s(k) + |R|)$  and  $(t(k) + |R|)$ . Thus, by the correctness of the extractability obfuscator  $e\mathcal{O}_{\{\mathcal{M}'_k\}}$ , for every  $k \in \mathbb{N}$ , every Turing machine  $M \in \mathcal{M}_k$ , and every valid witness  $w$  such that  $R(x, w) = 1$ , it holds that

$$\text{Evaluate}_{\{\mathcal{M}_k\}}(\text{Obfuscate}_{\{\mathcal{M}'_k\}}(1^k, M), w) = P_{R,x,m,M}(w) = M(w),$$

as desired.

*Succinctness of (Enc, Dec)*: Let  $p_s, p_t$  be the polynomials dictating the output size and runtime of the succinct obfuscation scheme, as in Definition ???. In particular, when obfuscating with respect to Turing machine class  $\{\mathcal{M}'_k\}$  defined above (with size and runtime bounded by  $(s(k) + |R|)$  and  $(t(k) + |R|)$ ), it holds that  $\text{Obfuscate}_{\{\mathcal{M}'_k\}}$  runs in time  $p_t(t(k) + |R|)$  and has output size  $p_s(s(k) + |R|)$ . By the construction of the FWE scheme, these values directly correspond to the runtime and ciphertext output size of the FWE encryption algorithm  $\text{Enc}_{R,\{\mathcal{M}_k\}}$ . Thus, succinctness of the FWE scheme follows directly, with equivalent polynomial bounds  $p_s, p_t$ .

*Security of (Enc, Dec)*: Fix any PPT FWE adversary  $\mathcal{A}$  and polynomial  $q(k)$ . By the construction of the FWE scheme, it is the case that  $\mathcal{A}$  is also directly a valid adversary in the extractability obfuscation security game. Thus, by the security of  $e\mathcal{O}$ , there exists a PPT extractor  $E$  and polynomial  $q(k)$  such that for every pair of Turing machines  $T_0, T_1 \in \mathcal{M}'_k$  and every auxiliary input  $z$ , if  $\mathcal{A}$  can distinguish between obfuscations  $\text{Obfuscate}_{\{\mathcal{M}'_k\}}(1^k, T_0)$  and  $\text{Obfuscate}_{\{\mathcal{M}'_k\}}(1^k, T_1)$  with advantage  $q(k)$ , then  $E$  successfully extracts an input  $w$  for which  $T_0(w) \neq T_1(w)$  with probability  $q'(k)$ . In particular, for any pair of messages  $m_0, m_1 \in \text{MSG}$ , this property holds for the pair of Turing machines  $P_{R,x,m_0,M}, P_{R,x,m_1,M} \in \mathcal{M}'_k$ . But, these obfuscations correspond directly to the distributions of FWE ciphertexts  $\text{Enc}(1^k, x, m_0, M)$  and  $\text{Enc}(1^k, x, m_1, M)$ . Thus, it follows that if  $\mathcal{A}$  succeeds in the FWE security game with advantage  $q(k)$  then  $E$  extracts a witness  $w$  for which  $P_{R,x,m_0,M}(w) \neq P_{R,x,m_1,M}(w)$  with probability  $q'(k)$ . Finally, note that on input  $w$  these two programs both run the same verification step checking if  $R(x, w) = 1$ . So if  $P_{R,x,m_0,M}(w) \neq P_{R,x,m_1,M}(w)$ , it must be that  $R(x, w) = 1$  holds (otherwise both programs output  $\perp$ ), and that  $M(m_0, w) \neq M(m_1, w)$ , as desired.

□



## 5 Applications to Functional Encryption

Recall the definition of (indistinguishability) Functional Encryption (FE), given in Section 2.2. We show how to use extractability obfuscation to directly achieve functional encryption for unbounded number of key queries and with full adaptive-message security for any unbounded size message space (without relying on complexity leveraging).

The intuition behind our scheme is simple. Let the public key of the FE scheme be the verification key for a signature scheme, and let the master secret key (needed to release secret keys  $sk_f$ ) be the signing key for the signature scheme. To encrypt a message  $m$ , obfuscate the program that on input  $f$  and a valid signature on  $f$  (given the public key) simply computes  $f(m)$ . The secret key  $sk_f$  for a function  $f$  is then simply the signature on  $f$ . (The high-level idea behind the construction is somewhat similar to the one used in [GKP<sup>+</sup>13], which uses witness encryption in combination with signature schemes to obtain simulation-based FE for a *single* function  $f$ ; in contrast, we here focus on FE for an unbounded number of functions).

Proving that this construction works is somewhat subtle. In fact, to make the proof go through, we require the signature scheme in use to be of a special *delegatable* kind—namely, we require the use of *functional signatures* [BGI13, BF13] (which can be constructed based on non-interactive zero-knowledge arguments of knowledge), which make it possible to delegate a signing key  $sk'$  that enables one to sign only messages that satisfy some predicate.<sup>10</sup> The delegation property is only used in the security reduction and, roughly speaking, makes it possible to simulate key queries without harming security for the messages selected by the attacker.

**Theorem 5.1.** *Assume the existence of non-interactive zero-knowledge arguments of knowledge (NIZKAoK) for  $NP$  and the existence of a extractability obfuscators for  $P/\text{poly}$ . Then there exists a (fully) indistinguishability-secure functional encryption scheme for arbitrary length messages.*

We first present the construction. Let  $(\text{FWE.Enc}, \text{FWE.Dec})$  be a FWE scheme for  $NP$  and  $P/\text{poly}$ , as in Definition 4.2. Recall by Theorem 4.3 this is equivalent to an extractability obfuscator for  $P/\text{poly}$ . Let  $(\text{Sig.Setup}, \text{Sig.KeyGen}, \text{Sig.Sign}, \text{Sig.Verify})$  be a succinct functional signature scheme for  $P/\text{poly}$ , as guaranteed by Theorem 2.7 based on NIZKAoK. Consider the following tuple of algorithms.

- $\text{FE.Setup}(1^k)$ : On input the security parameter  $1^k$ ,  $\text{FE.Setup}$  samples a key pair  $(\text{msk}_{\text{sig}}, \text{vk}) \leftarrow \text{Sig.Setup}(1^k)$  for the signature scheme, and generates a key  $\text{sk}_1 \leftarrow \text{Sig.KeyGen}(\text{msk}, 1)$  that allows signing all messages (i.e., always-accepting predicate  $1(M) = M \forall M$ ). It outputs  $\text{pp} = \text{vk}$  and  $\text{msk} = (\text{sk}_1, \text{vk})$ .
- $\text{FE.KeyGen}(\text{msk}, M)$ : On input the master secret key  $\text{msk} = (\text{sk}_1, \text{vk})$  and description of a Turing machine  $M$ ,  $\text{FE.KeyGen}$  generates a signature on  $M$  via  $\sigma_M \leftarrow \text{Sig.Sign}(\text{sk}_1, M)$ . It outputs  $\text{sk}_M := \sigma_M$ .
- $\text{FE.Enc}(\text{pp}, m)$ : On input the public parameters  $\text{pp} = \text{vk}$  and message  $m$ ,  $\text{FE.Enc}$  does the following. Define the  $NP$  relation  $R_{\text{vk}}$  such that  $R_{\text{vk}}(x, (w_f, w_\sigma, p(k))) = 1$  iff  $\text{Verify}(\text{vk}, w_f, w_\sigma) = 1$  (i.e., a message-signature pair  $w_f, w_\sigma$  with respect to  $\text{vk}$ , together with an arbitrary polynomial  $p(k)$ , yield a valid witness for any statement  $x$ ).

Let  $t(k)$  be a time bound on the desired class of supported Turing machines  $\{\mathcal{M}_k\}$ , and let  $\{\mathcal{U}_k\}$  be the class of universal Turing machines that evaluate an input TM for  $t(k)$  steps. Denote by  $U \in \mathcal{U}_k$  the Turing machine whose input is composed of: a message  $m$ , a Turing

---

<sup>10</sup>Note that functional signatures were not needed in [GKP<sup>+</sup>13], as they only consider a single key query. In our case, functional signatures are needed to answer “CCA”-type key queries.

machine description  $M$ , a string  $\sigma$ , and a polynomial  $t_M(k)$ ; and which evaluates the Turing machine  $M$  on input  $m$  for  $\min\{t(k), t_M(k)\}$  steps.

Output an FWE encryption of message  $m$  with respect to an arbitrary statement  $x = 0$  and the Turing machine  $U \in \{\mathcal{U}_k\}$ :

$$c \leftarrow \text{FWE.Enc}_{R_{\text{vk}}, \{\mathcal{U}_k\}}(1^k, 0, m, U).$$

- $\text{FE.Dec}(\text{sk}_M, c)$ : On input a secret key  $\text{sk}_M = \sigma_M$  and ciphertext  $c$ , output the FWE decryption of  $c$  using witness  $(M, \sigma_M, t_M(k))$ , where  $t_M(k)$  is a runtime bound on  $M$ . That is, evaluate  $\text{FWE.Dec}_{R_{\text{vk}}, \{\mathcal{U}_k\}}(c, (M, \sigma_M, t_M(k)))$ .

*Proof.* We analyze the correctness, security, and ciphertext/key sizes of the constructed scheme.

**Correctness.** Follows by the correctness of the FWE and functional signature schemes. Namely, given any signature  $\sigma_M$  on a Turing machine  $M$ , the tuple  $(M, \sigma_M, t_M(k))$  will be a valid witness for the statement  $x = 0$  with respect to  $R_{\text{vk}}$ , and thus for properly generated ciphertext  $\text{FWE.Enc}_{R_{\text{vk}}, \{\mathcal{U}_k\}}(1^k, 0, m, U)$ , the output of decryption  $\text{FWE.Dec}_{R_{\text{vk}}, \{\mathcal{U}_k\}}(c, (M, \sigma_M, t_M(k))) = U(m, (M, \sigma_M, t_M(k))) = M(m)$  by construction.

**Security.** Let  $\mathcal{A}$  be a PPT functional encryption adversary, and let  $Q(k)$  be an upper bound on the number of key queries made by  $\mathcal{A}$  during the FE security game.

At a high level, the proof of security will follow three steps: First, we argue that  $\mathcal{A}$ 's distinguishing advantage in the FE security game cannot decrease by too much if we instead answer his post-challenge key queries using a *restricted* signing key  $\text{sk}_{P_{\text{eq}}}$ , which only allows signing messages corresponding to Turing machines  $M$  for which  $M(m_0) = M(m_1)$  (where  $m_0, m_1$  are the selected challenge messages). This will hold by the function privacy property of the functional signature scheme. Next, we show that by the security of the FWE, any such adversary who succeeds in distinguishing a ciphertext of  $m_0$  versus  $m_1$  with noticeable probability within this game implies the existence of an extractor  $E$  who can efficiently find a witness  $w = (M', \sigma_{M'}, \ell(k))$  (for the relation  $R_{\text{vk}}$ ) for which  $U(m_0, (M', \sigma_{M'}, \ell(k))) \neq U(m_1, (M', \sigma_{M'}, \ell(k)))$ ; in particular,  $\sigma_{M'}$  must be a valid signature on some machine  $M'$  for which  $M'(m_0) \neq M'(m_1)$ . Finally, we demonstrate that such an extractor can be used to produce a forgery in the functional signature scheme, providing a contradiction.

We proceed with the first step. Consider the following hybrid experiments:

**Hybrid 0.** The standard functional encryption security game. Namely,

1. The adversary  $\mathcal{A}$  receives public parameters  $\text{pp}$  for the FE scheme, where  $(\text{pp}, \text{msk}) \leftarrow \text{FE.Setup}(1^k)$ . Recall  $\text{msk}$  consists of a signing key  $\text{sk}_1$  that enables signing all messages.
2.  $\mathcal{A}$  adaptively makes key queries for Turing machines  $M$ , and for each receives  $\sigma_M \leftarrow \text{Sig.Sign}(\text{sk}_1, M)$ .
3. After some number of queries,  $\mathcal{A}$  outputs a pair of messages  $(m_0, m_1)$  for which  $M(m_0) = M(m_1)$  for each queried  $M$ . The FE challenger samples a random bit  $b \leftarrow \{0, 1\}$ , generates an encryption of  $m_b$  via  $c \leftarrow \text{FWE.Enc}_{R_{\text{vk}}, \{\mathcal{U}_k\}}(1^k, 0, m_b, U)$ , and sends  $c$  to  $\mathcal{A}$  as the challenge ciphertext.
4.  $\mathcal{A}$  may continue to adaptively make key queries for programs  $M$ , with the restriction that  $M(m_0) = M(m_1)$ . Each query is answered as above.
5. Eventually,  $\mathcal{A}$  outputs a guess  $b'$  for the bit  $b$ .

**Hybrids**  $i = 1, \dots, Q$ . Same as the previous hybrid, except that the first  $i$  post-challenge key queries are answered with respect to a *restricted* signing key  $\text{sk}_{P_{\text{eq}}}$  for the function (predicate)  $P_{\text{eq}}$  that allows one to sign exactly Turing machine descriptions  $M$  for which  $M(m_0) = M(m_1)$ . Namely,

1.-3. Identical to Hybrid 0.

4. The FE challenger generates a *limited* signing key  $\text{sk}_{P_{\text{eq}}} \leftarrow \text{Sig.KeyGen}(\text{msk}, P_{\text{eq}})$ .  $\mathcal{A}$  may continue to adaptively make key queries for programs  $M$ , with the restriction that  $M(m_0) = M(m_1)$ .

The first  $i$  such queries are answered using this *restricted* key:  $\sigma_M \leftarrow \text{Sig.Sign}(\text{sk}_{P_{\text{eq}}}, M)$ . All remaining queries are answered using the standard key:  $\sigma_M \leftarrow \text{Sig.Sign}(\text{sk}_1, M)$ .

5. Identical to Hybrid 0.

For  $i = 0, \dots, Q$ , denote by  $\text{adv}_i^z$  the advantage of  $\mathcal{A}$  in guessing the bit  $b$  in Hybrid  $i$  on auxiliary input  $z$ , (as a function of the security parameter  $k$ ). The theorem follows from the following sequence of claims:

**Claim 5.2.** *There exists a negligible function  $\nu(k)$  such that for each  $i \in [Q]$ , and for any auxiliary input  $z$ ,  $\text{adv}_i^z \geq \text{adv}_{i-1}^z - \nu(k)$ .*

*Proof.* This claim will hold by the function privacy property of the functional signature scheme. Namely, for each  $i \in [Q]$  consider the following adversary:

$\mathcal{A}_{\text{priv}}^i(1^K, z)$ :

1.  $\mathcal{A}_{\text{priv}}^i$  is given keys  $(\text{vk}, \text{msk}) \leftarrow \text{Sig.Setup}(1^k)$  in the function privacy challenge.
2.  $\mathcal{A}_{\text{priv}}^i$  submits the all-accepting function  $1(M) \equiv M$  as the first of his two challenge functions, and receives a corresponding signing key  $\text{sk}_1 \leftarrow \text{Sig.KeyGen}(\text{msk}, 1)$ .
3.  $\mathcal{A}_{\text{priv}}^i$  simulates interaction with the functional encryption adversary  $\mathcal{A}$ . First, he forwards  $\text{vk}$  to  $\mathcal{A}$  as the public parameters of the FE scheme. For each key query  $M$  made by  $\mathcal{A}$ , the adversary  $\mathcal{A}_{\text{priv}}^i$  generates a signature on  $M$  using the key  $\text{sk}_1$ : i.e.,  $\sigma_M \leftarrow \text{Sig.Sign}(\text{sk}_1, M)$ .
4. Eventually,  $\mathcal{A}$  outputs a pair of messages  $m_0, m_1$ .  $\mathcal{A}_{\text{priv}}^i$  generates a challenge ciphertext in the FE game by sampling a random bit  $b \leftarrow \{0, 1\}$  and encrypting  $c \leftarrow \text{FWE.Enc}_{R_{\text{vk}}, \{U_k\}}(1^k, 0, m_b, U)$ .  $\mathcal{A}_{\text{priv}}^i$  sends  $c$  to  $\mathcal{A}$ .
5.  $\mathcal{A}_{\text{priv}}^i$  submits as his second challenge function  $P_{\text{eq}}$  defined by  $P_{\text{eq}}(M) = M$  if  $M(m_0) = M(m_1)$  and  $= \perp$  otherwise. He receives a corresponding signing key  $\text{sk}_{P_{\text{eq}}} \leftarrow \text{Sig.KeyGen}(\text{msk}, P_{\text{eq}})$ .
6.  $\mathcal{A}_{\text{priv}}^i$  now simulates interaction with  $\mathcal{A}$  as follows. (Note that any queried  $M$  for which  $M(m_0) \neq M(m_1)$  is ignored).

For the first  $i - 1$  of  $\mathcal{A}$ 's post-challenge key queries  $M$ ,  $\mathcal{A}_{\text{priv}}^i$  generates a signature *using key*  $\text{sk}_{P_{\text{eq}}}$ : i.e.,  $\sigma_M \leftarrow \text{Sig.Sign}(\text{sk}_{P_{\text{eq}}}, M)$ .

For  $\mathcal{A}$ 's  $i$ th post-challenge query,  $\mathcal{A}_{\text{priv}}^i$  submits the pair of preimages  $(M, M)$  to the function privacy challenger (note that  $1(M) = P_{\text{eq}}(M) = M$ ), and receives a signature  $\sigma_M$  generated *either* using key  $\text{sk}_1$  or key  $\text{sk}_{P_{\text{eq}}}$ .

For  $\mathcal{A}$ 's remaining post-challenge queries,  $\mathcal{A}_{\text{priv}}^i$  generates a signature using key  $\text{sk}_1$ : i.e.,  $\sigma_M \leftarrow \text{Sig.Sign}(\text{sk}_1, M)$ .

7. Eventually  $\mathcal{A}$  outputs a bit  $b'$ . If  $b' = b$  correctly guesses the bit sampled in Step 4, then  $\mathcal{A}_{\text{priv}}^i$  outputs 1; otherwise,  $\mathcal{A}_{\text{priv}}^i$  outputs  $P_{\text{eq}}$ .

Note that if the function privacy challenger selected the function 1, then  $\mathcal{A}_{\text{priv}}^i$  perfectly simulates Hybrid  $i-1$ , and if the challenger selected the function  $P_{\text{eq}}$ , then  $\mathcal{A}_{\text{priv}}^i$  perfectly simulates Hybrid  $i$ . Thus,  $\mathcal{A}_{\text{priv}}^i$ 's advantage in the function privacy game is exactly equal to the difference  $\text{adv}_i^z - \text{adv}_{i-1}^z$ . By the function privacy property of the functional signature scheme, it thus follows that this difference is negligible.  $\square$

Next, we use the FWE security to show that any successful distinguishing adversary in this Hybrid  $Q$  experiment implies an extractor who finds a witness for the statement  $x = 0$  with respect to the relation  $R_{\text{vk}}$ , for which the FWE function evaluations on  $m_0$  and  $m_1$  differ.

**Claim 5.3.** *Suppose there exists auxiliary input  $z_{\mathcal{A}}$  and polynomial  $p(k)$  for which  $\text{adv}_Q^{z_{\mathcal{A}}} \geq 2/p(k)$ . Then there exists a PPT extractor  $E$ , a polynomial  $q(k)$ , and efficiently samplable distribution  $\mathcal{D}$  such that with probability  $1/p(k)$  over  $(\text{vk}, m_0, m_1, z_E) \leftarrow \mathcal{D}$  it holds that*

$$\Pr \left[ w \leftarrow E(1^k, m_0, m_1, z_E) : R_{\text{vk}}(x = 0, w) = 1 \wedge U(m_0, w) \neq U(m_1, w) \right] \geq \frac{1}{q(k)}.$$

*Proof.* Define the following distribution  $\mathcal{D}$ , as a function of  $\mathcal{A}$  and  $z_{\mathcal{A}}$ :

Distribution  $\mathcal{D}$ :

1. Sample a key pair for the functional signature scheme  $(\text{vk}, \text{msk}) \leftarrow \text{Sig.Setup}(1^k)$ , and generate a signing key for the “all-accepting” function  $1(M) \equiv M$ , by  $\text{sk}_1 \leftarrow \text{Sig.KeyGen}(\text{msk}, 1)$ .
2. Simulate the action of  $\mathcal{A}$  within the Hybrid  $Q$  experiment on auxiliary input  $z_{\mathcal{A}}$ . Namely, forward the public parameters  $\text{pp} := \text{vk}$  to  $\mathcal{A}$ , and answer each of his (adaptive) key queries  $M$  by producing a signature on  $M$  using key  $\text{sk}_1$ .
3. At some point,  $\mathcal{A}$  outputs a pair of messages  $m_0, m_1$  for the Hybrid  $Q$  FE ciphertext challenge. Denote by  $\text{view}_{\mathcal{A}}$  the current complete view of the adversary  $\mathcal{A}$  up to this point of the simulation (which will enable one to return  $\mathcal{A}$  to this state and continue simulation consistently).
4. Generate a signing key for the function  $P_{\text{eq}}$  defined by  $P_{\text{eq}}(M) = M$  if  $M(m_0) = M(m_1)$  and  $= \perp$  otherwise. Namely,  $\text{sk}_{P_{\text{eq}}} \leftarrow \text{Sig.KeyGen}(\text{msk}, P_{\text{eq}})$ .
5. Output the tuple  $(\text{vk}, m_0, m_1, z_E = (z_{\mathcal{A}}, \text{view}_{\mathcal{A}}, \text{sk}_{P_{\text{eq}}}))$ .

We now define an adversary  $\mathcal{A}_{\text{FWE}}$  for the FWE scheme, as a function of  $\mathcal{A}$ . Given a tuple of values  $(\text{vk}, m_0, m_1, z_E = (z_{\mathcal{A}}, \text{view}_{\mathcal{A}}, \text{sk}_{P_{\text{eq}}}))$  from the support of  $\mathcal{D}$ , and a ciphertext  $c$ ,  $\mathcal{A}_{\text{FWE}}$  does the following:

Adversary  $\mathcal{A}_{\text{FWE}}(1^k, \text{vk}, m_0, m_1, z_E, c)$ :

1. Using  $\text{view}_{\mathcal{A}}$ , return  $\mathcal{A}$  to the same state of execution as in the corresponding earlier simulation during the  $\mathcal{D}$  sampling process.
2. Simulate the actions of  $\mathcal{A}$  upon receiving challenge ciphertext  $c$ . For each subsequent key query  $M$  made by  $\mathcal{A}$ , answer by producing a signature on  $M$  using key  $\text{sk}_{P_{\text{eq}}}$ .
3. Eventually,  $\mathcal{A}$  outputs a bit guess  $b'$  for the challenge ciphertext.
4. Output the bit  $b$ .

Note that the interaction with the adversary  $\mathcal{A}$  in sampling from  $\mathcal{D}$  is precisely a simulation of Steps 1-3 in the Hybrid  $Q$  experiment (*except* the challenge ciphertext generation), and the interaction with  $\mathcal{A}$  made by  $\mathcal{A}_{\text{FWE}}$  is precisely a simulation of the remaining Steps 4-5 of Hybrid  $Q$ .

We are assuming  $\text{adv}_Q^{z_A} \geq 2/p(k)$  (i.e., the adversary  $\mathcal{A}$  distinguishes challenge ciphertexts in Hybrid  $Q$  with noticeable advantage  $2/p(k)$ ). That is,

$$\Pr \left[ (\text{vk}, m_0, m_1, z_E) \leftarrow \mathcal{D}(\mathcal{A}, z_A); b \leftarrow \{0, 1\}; c \leftarrow \text{FWE.Enc}_{R_{\text{vk}}, \{\mathcal{U}_k\}}(1^k, 0, m_b, U); \right. \\ \left. b' \leftarrow \mathcal{A}_{\text{FWE}}(1^k, \text{vk}, m_0, m_1, z_E, c) : b = b' \right] \geq \frac{1}{2} + \frac{2}{p(k)}.$$

This implies that with probability at least  $1/p(k)$  over  $(\text{vk}, m_0, m_1, z_E) \leftarrow \mathcal{D}$ , we have

$$\Pr \left[ b \leftarrow \{0, 1\}; c \leftarrow \text{FWE.Enc}_{R_{\text{vk}}, \{\mathcal{U}_k\}}(1^k, 0, m_b, U); b' \leftarrow \mathcal{A}_{\text{FWE}}(1^k, \text{vk}, m_0, m_1, z_E) : b' = b \right] \geq \frac{1}{2} + \frac{1}{p(k)}.$$

But, this says exactly that the adversary  $\mathcal{A}_{\text{FWE}}$  succeeds in distinguishing FWE ciphertexts with noticeable advantage  $1/p(k)$ . Therefore, by the security of the FWE scheme, there exists a corresponding extractor  $E$  and polynomial  $q(k)$  such that, with probability  $1/p(k)$  over  $(\text{vk}, m_0, m_1, z_E) \leftarrow \mathcal{D}$  it holds that

$$\Pr \left[ w \leftarrow E(1^k, m_0, m_1, z_E) : R_{\text{vk}}(x = 0, w) = 1 \wedge U(m_0, w) \neq U(m_1, w) \right] \geq \frac{1}{q(k)},$$

as desired.  $\square$

Finally, we prove that such an extractor cannot exist, as it would break the unforgeability of the underlying functional signature scheme.

**Claim 5.4.** *There cannot exist a PPT algorithm  $E$  as in Claim 5.3.*

*Proof.* Suppose, to the contrary, such a PPT  $E$  exists. We will use  $E$  to construct an adversary  $\mathcal{A}_{\text{sig}}$  who breaks the unforgeability of the underlying functional signature scheme. At a high level,  $\mathcal{A}_{\text{sig}}$  proceeds as follows. First, he will use his functional signature oracles  $O_{\text{sign}}$  and  $O_{\text{key}}$  to simulate interaction with  $\mathcal{A}$  in the pre-challenge portion of the Hybrid  $Q$  interaction, and then to request the special key  $\text{sk}_{P_{\text{eq}}}$ . Together, this will allow  $\mathcal{A}_{\text{sig}}$  to sample a tuple according to the distribution  $\mathcal{D}$ . Then, given this tuple,  $\mathcal{A}_{\text{sig}}$  can simply execute the extractor algorithm  $E$  to produce (with noticeable probability) a witness  $w$  for the relation  $R_{\text{vk}}$  for which  $U(m_0, w) \neq U(m_1, w)$ . But, recalling the choice of Turing machine  $U$  and relation  $R_{\text{vk}}$ , it holds that a witness  $w$  contains a Turing machine description  $M'$  and signature  $\sigma_{M'}$  on  $M'$  with respect to  $\text{vk}$ . We will then argue that this pair  $(M', \sigma_{M'})$  yields a forgery in the functional signature game.

Formally, consider the following adversary  $\mathcal{A}_{\text{sig}}$ .

Adversary  $\mathcal{A}_{\text{sig}}(1^k, z_A)$ :

1.  $\mathcal{A}_{\text{sig}}$  produces a sample from the distribution  $\mathcal{D}$ , as follows:
  - (a)  $\mathcal{A}_{\text{sig}}$  receives a verification key  $\text{vk}$  for the functional signature challenge.
  - (b) Simulate the action of  $\mathcal{A}$  within the Hybrid  $Q$  experiment on auxiliary input  $z_A$ . First, forward the public parameters  $\text{pp} := \text{vk}$  to  $\mathcal{A}$ . For each FE key query  $M$  made by  $\mathcal{A}$ ,  $\mathcal{A}_{\text{sig}}$  makes a query to his signing oracle  $\sigma_M \leftarrow O_{\text{sign}}(1, 1, M)$ ; i.e., for function  $1(M) \equiv M$ , consistent index  $i = 1$  (so that all queries are answered with the same key  $\text{sk}_1$ ), on the message  $M$ . Send  $\sigma_M$  to  $\mathcal{A}$  as the response to his key generation query.
  - (c) At some point  $\mathcal{A}$  outputs a pair of messages  $m_0, m_1$ . Denote by  $\text{view}_{\mathcal{A}}$  the view of  $\mathcal{A}$  up to this point in the simulation.

- (d)  $\mathcal{A}_{\text{sig}}$  queries his signing key oracle  $O_{\text{key}}(P_{\text{eq}})$  on the function  $P_{\text{eq}}$  defined by  $P_{\text{eq}}(M) = M$  if  $M(m_0) = M(m_1)$  and  $\perp$  otherwise. Denote the resulting key by  $\text{sk}_{P_{\text{eq}}}$ .
- (e) Output the tuple  $(\text{vk}, m_0, m_1, z_E = (z_{\mathcal{A}}, \text{view}_{\mathcal{A}}, \text{sk}_{P_{\text{eq}}}))$  as the sample from  $\mathcal{D}$ .

Note that by construction this tuple indeed has the correct distribution  $\mathcal{D}$ .

2.  $\mathcal{A}_{\text{sig}}$  then executes the extractor algorithm  $E$  on input  $(1^k, m_0, m_1, z_E)$ . Denote the output by  $w = (M', \sigma_{M'}, \ell(k))$ .
3.  $\mathcal{A}_{\text{sig}}$  outputs the pair  $(M', \sigma_{M'})$  as his forgery in the functional signature scheme.

Note that since the tuple  $(\text{vk}, m_0, m_1, z_E)$  generated by  $\mathcal{A}_{\text{sig}}$  in Step 1 is distributed according to  $\mathcal{D}$ , we have by Claim 5.3 that with probability  $1/p(k)$  over this sampling, it holds that  $E(1^k, m_0, m_1, z_E)$  succeeds in producing a witness  $w = (M', \sigma_{M'}, \ell(k))$  for which  $R_{\text{vk}}(x = 0, w) = 1$  and that  $U(m_0, w) \neq U(m_1, w)$  with probability  $1/p(k)$ .

Now, recall the choice of relation  $R_{\text{vk}}$  and Turing machine  $U$ . A tuple  $(M', \sigma_{M'}, \ell(k))$  is a witness for the statement  $x = 0$  with respect to  $R_{\text{vk}}$  if and only if  $\sigma_{M'}$  is a valid signature on  $M'$  with respect to  $\text{vk}$ : that is,  $\text{Sig.Verify}(\text{vk}, M', \sigma_{M'}) = 1$ . It remains to show that  $M'$  is not covered by any of  $\mathcal{A}_{\text{sig}}$ 's key generation/signing oracle queries in the function signature game (so that the pair  $(M', \sigma_{M'})$  indeed constitutes a forgery). Recall the Turing machine  $U$  takes as input a message  $m$ , a Turing machine description  $M'$ , a signature  $\sigma_{M'}$ , and polynomial  $\ell(k)$ , and outputs the evaluation of  $M'$  on input  $m$  (executed for  $\ell(k)$  steps). Thus,  $U(m_0, (M', \sigma_{M'}, \ell(k))) \neq U(m_1, (M', \sigma_{M'}, \ell(k)))$  means that  $M(m_0) \neq M(m_1)$ . But, in the functional encryption security game, *all* key generation queries  $M$  made by the adversary  $\mathcal{A}$  must satisfy  $M(m_0) = M(m_1)$  (otherwise either the pair of messages  $(m_0, m_1)$  or the query  $M$  would have been deemed invalid). Further, by definition it holds that  $P_{\text{eq}}(M') = \perp$ , so that the queried key  $\text{sk}_{P_{\text{eq}}}$  does not enable signing  $M'$ .

Thus, we have that with probability  $(1/p(k))(1/q(k))$ , the adversary  $\mathcal{A}_{\text{sig}}$  produces a valid forgery  $(M', \sigma_{M'})$  in the functional signature scheme, contradicting its assumed security. □

Security of the constructed FE scheme follows.

**Efficiency Analysis.** Let  $s(k), t(k)$  be polynomial bounds on the size and runtime of TMs in the supported class of Turing machines  $\{\mathcal{M}_k\}$ .

- Setup: Sampling a verification key for the functional signature scheme takes time  $\text{poly}(k)$ , and results in public parameters of size  $\text{poly}(k)$ .
- KeyGen: A secret key  $\text{sk}_M$  for Turing machine  $M$  is a signature on  $M$ , which is of size  $\text{poly}(k)$ .
- Encryption: An encryption of a message is an FWE encryption of  $m$ , together with Turing machine  $U$  (defined in FE.Enc), performed with respect to the  $NP$  relation  $R_{\text{vk}}$  and class of universal Turing machines  $\{\mathcal{U}_k\}$  that execute an input TM for a maximum of  $t(k)$  steps (note that this Turing machine class has size bound  $s_{\mathcal{U}}(k) = O(1)$ ).  $R_{\text{vk}}$  corresponds to verifying a signature on a Turing machine from the class  $\{\mathcal{M}_k\}$ , which takes time proportional to  $s(k)$ . Thus, by succinctness of the FWE scheme, the corresponding ciphertext size is  $\text{poly}(k, |R_{\text{vk}}|, s_{\mathcal{U}}(k)) = \text{poly}(k, s(k))$ .
- Decryption: Takes time  $\text{poly}(k, s(k), t(k))$ .

□

## 6 Relating Extractability and Indistinguishability Obfuscation

A natural question is whether we can obtain extractability obfuscation from indistinguishability obfuscation. We address this question in two different settings: first directly in the context of obfuscation, and second in the language of FWE. (Recall that these two notions are equivalent when dealing with arbitrary circuits and arbitrary functions; however, when considering restricted function classes, there are interesting differences).

In Section 6.1, we demonstrate that any indistinguishability obfuscation in fact implies a weak version of extractability obfuscation, in which extraction is only guaranteed when the two circuits differ on only polynomially many inputs. In Section 6.2, we define a weaker notion of FWE mirroring the definition of indistinguishability obfuscation, and provide a transformation from any such indistinguishability FWE to standard FWE for languages with polynomially many witnesses.

The two results are incomparable, in that the former transformation (in Section 6.1) starts with a stronger assumption and yields a stronger result. Indeed, if one begins with indistinguishability FWE for all NP and  $P/\text{poly}$ , then by the equivalence of FWE and obfuscation, the former transformation yields a stronger outcome in the setting of FWE, guaranteeing indistinguishability of encryptions of messages  $m_0, m_1$  with respect to a function  $f$  and NP statement  $x$  with potentially exponentially many witnesses, as long as only *polynomially many such witnesses  $w$  produce differing outputs*  $f(m_0, w) \neq f(m_1, w)$ . On the other hand, the FWE transformation (in Section 6.2) also treats the case of restricted function classes. For example, it provides a method for transforming indistinguishability FWE for the trivial function  $f(m, w) = m$  to FWE for the same function  $f$ . It is easy to see that indistinguishability FWE for this particular  $f$  is equivalent to the notion of witness encryption [GGSW13], and FWE for the same  $f$  is equivalent to the notion of extractable witness encryption of [GKP<sup>+</sup>13]. The transformation in Section 6.2 thus shows how to turn witness encryption to extractable witness encryption for the case of languages with polynomially many witness.

### 6.1 From Indistinguishability Obfuscation to Extractability Obfuscation for Circuits with Polynomial Differing Inputs

We show that indistinguishability obfuscation directly implies a weak version of extraction obfuscation, where extraction is successful for any pair of circuits  $C_0, C_1$  that vary on polynomially many inputs.

**Definition 6.1** (Weak Extractability Obfuscation). A uniform transformation  $\mathcal{O}$  is a *weak extractability obfuscator* for a class of Turing machines  $\mathcal{M} = \{\mathcal{M}_k\}$  if the following holds. For every PPT adversary  $\mathcal{A}$  and polynomial  $p(k)$ , there exists a PPT algorithm  $E$  and polynomials  $p_E(k), t_E(k)$  for which the following holds. For every polynomial  $d(k)$ , for all sufficiently large  $k$ , and every pair of circuits  $M_0, M_1 \in \mathcal{M}_k$  *differing on at most  $d(k)$  inputs*, and every auxiliary input  $z$ ,

$$\begin{aligned} \Pr \left[ b \leftarrow \{0, 1\}; \tilde{M} \leftarrow \mathcal{O}(1^k, C_b) : \mathcal{A}(1^k, \tilde{M}, M_0, M_1, z) = b \right] &\geq \frac{1}{2} + \frac{1}{p(k)} \\ \implies \Pr \left[ x \leftarrow E(1^k, M_0, M_1, z) : M_0(x) \neq M_1(x) \right] &\geq \frac{1}{p_E(k)}, \end{aligned}$$

and the runtime of  $E$  is  $t_E(k, d(k))$ .

**Theorem 6.2.** *Let  $\mathcal{O}$  be an indistinguishability obfuscator for  $P/\text{poly}$ . Then  $\mathcal{O}$  is also a weak extractability obfuscator for  $P/\text{poly}$ .*

Denote by  $n = n(k)$  the (polynomial) input length of the circuits in question. At a high level, the extractor  $E$  associated with an adversary  $\mathcal{A}$  performs a form of binary search over  $\{0, 1\}^n$  for a desired input by considering a sequence of intermediate circuits lying “in between”  $C_0$  and  $C_1$ . The goal is that after  $n$  iterations,  $E$  will reach a pair of circuits  $C^{\text{Left}}, C^{\text{Right}}$  for which: (1)  $\mathcal{A}$  can still distinguish between obfuscations  $\{\mathcal{O}(C^{\text{Left}})\}$  and  $\{\mathcal{O}(C^{\text{Right}})\}$ , and yet (2)  $C^{\text{Left}}$  and  $C^{\text{Right}}$  are identical on all inputs except a single known  $x$ , for which  $C^{\text{Left}}(x) = C_0(x)$  and  $C^{\text{Right}}(x) = C_1(x)$ . Thus, by the indistinguishability security of  $\mathcal{O}$ , it must be that  $E$  has extracted an input  $x$  for which  $C_0(x) \neq C_1(x)$ .

To demonstrate, consider the case where the circuits  $C_0, C_1$  differ on a single unknown input  $x^*$ . In the first step, the extractor algorithm  $E$  will consider an intermediate circuit  $C^{\text{Mid}}$  equal to  $C_0$  on the first half of its inputs, and equal to  $C_1$  on the second half of its inputs. Then since  $C^{\text{Mid}}(x^*) \in \{C_0(x^*), C_1(x^*)\}$  and all three circuits agree on inputs  $x \neq x^*$ , it must be that  $C^{\text{Mid}}$  is *equivalent* to either  $C_0$  or  $C_1$ . By the security of the indistinguishability obfuscator, it follows that the obfuscations of such equivalent circuits are indistinguishable. But, if an adversary  $\mathcal{A}$  distinguishes between obfuscations of  $C_0$  and  $C_1$  with noticeable advantage  $\epsilon$ , then  $\mathcal{A}$  must successfully distinguish between obfuscations of  $C_0$  &  $C^{\text{Mid}}$  or  $C^{\text{Mid}}$  &  $C_1$ . Namely, it must be the case that  $\mathcal{A}$ 's distinguishing advantage is very small between one of these pairs of distributions (corresponding to the case  $C^{\text{Mid}} \equiv C_b$ ) and is nearly  $\epsilon$  for the other pair of distributions (corresponding to  $C^{\text{Mid}} \not\equiv C_{1-b}$ ). Thus, by generating samples from these distributions and estimating  $\mathcal{A}$ 's distinguishing advantages for the two distribution pairs,  $E$  can determine whether  $C^{\text{Mid}} \equiv C_0$  or  $C^{\text{Mid}} \equiv C_1$  and, in turn, has learned whether  $x^*$  lies in the first or second half of the input space. This process is then repeated iteratively within a smaller window (i.e., considering a new intermediate circuit lying “in between”  $C^{\text{Mid}}$  and  $C_b$  for which  $C^{\text{Mid}} \not\equiv C_b$ ). In each step, we decrease the input space by a factor of two, until  $x^*$  is completely determined.

The picture becomes more complicated, however, when there are several inputs on which  $C_0$  and  $C_1$  disagree. Here the intermediate circuit  $C^{\text{Mid}}$  need not agree with either  $C^{\text{Left}}$  or  $C^{\text{Right}}$  on all inputs. Thus, whereas above  $\mathcal{A}$ 's distinguishing advantage along one of the two paths was guaranteed to drop no more than a negligible amount, here in each step  $\mathcal{A}$ 's advantage could split by as much as half. At this rate, after only  $\log k$  iterations,  $\mathcal{A}$ 's advantage will drop below usable levels, and the binary search approach will fail. Indeed, if  $C_0, C_1$  differ on superpolynomially many inputs  $d(k) \in k^{\omega(1)}$ , there may not even *exist* a pair of adjacent circuits  $C^{\text{Left}}$  and  $C^{\text{Right}}$  satisfying the desired properties (1) and (2) described above. (Intuitively, for example, it could be the case that each time one evaluation is changed from  $C_0(x)$  to  $C_1(x)$ , the adversary's probability of outputting 1 increases by the negligible amount  $1/d$ ).

We show, however, that if there are polynomially many differing inputs  $D \subset \{0, 1\}^n$  for which  $C_0(x) \neq C_1(x)$ , then this issue can be overcome. The key insight is that, in any step of the binary search where the adversary's distinguishing advantage may split noticeably among the two possible continuing paths, this step must also split the *set of differing inputs* into two subsets: that is, the number of points  $d'$  on which  $C^{\text{Left}}$  and  $C^{\text{Right}}$  disagree is equal to the *sum* of the number of points  $d^L$  on which  $C^{\text{Mid}}$  and  $C^{\text{Left}}$  disagree and the number of points  $d^R$  on which  $C^{\text{Mid}}$  and  $C^{\text{Right}}$  disagree. Then even though the adversary's distinguishing advantage may split as  $\epsilon' = \epsilon^L + \epsilon^R$ , for at least one of the two paths  $b \in \{L, R\}$ , it must be that the ratio of  $\epsilon^b/d^b \geq \epsilon'/d'$  is roughly maintained (up to a negligible amount). Since there are only polynomially many total disagreeing inputs  $d(k) \in k^{O(1)}$  to start, and assuming  $\mathcal{A}$  begins with noticeable distinguishing advantage, the original ratio  $\epsilon/d$  at the root node begins as a noticeable amount. And so we are guaranteed that there exists a path down the tree for which  $\epsilon'/d'$  (and, in particular, the intermediate distinguishing advantage  $\epsilon'$ ) stays above this noticeable amount  $\epsilon/d$ . Our extractor  $E$  will find this path by simply



following *all paths* which maintain distinguishing advantage above this value. By the security of the indistinguishability obfuscation scheme, there will be at most polynomially many such paths (corresponding to those terminating at the special inputs  $x \in D$ ), and all other paths in the tree will be pruned.

More specifically, our extractor algorithm  $E$  runs as follows. At the beginning of execution, it sets a fixed threshold  $\text{thresh} = \epsilon/dk$  based on the original (signed) distinguishing advantage  $\epsilon$  of  $\mathcal{A}$  and the number of inputs  $d$  on which the circuits differ (note that if this value  $d = d(k)$  is unknown,  $E$  will repeat the whole extraction procedure with guesses  $k, k^2, k^{2^2}, k^{2^3}$ , etc, for this value). At each step it considers three circuits  $C^{\text{Left}}, C^{\text{Mid}}, C^{\text{Right}}$ , and estimates  $\mathcal{A}$ 's (signed) distinguishing advantage between obfuscations of  $C^{\text{Left}}$  &  $C^{\text{Mid}}$  and of  $C^{\text{Mid}}$  &  $C^{\text{Right}}$ , using repeated sampling with sufficiently low error ( $\text{err} = \epsilon/dk^2$ ). For each pair that yields distinguishing probability above  $\text{thresh}$  (which could be neither, one, or both pairs),  $E$  recurses by repeating this process at a circuit lying between the relevant window. More explicitly, if the left pair yields sufficient distinguishing advantage, then  $E$  will repeat the process for the triple of circuits  $C^{\text{Left}}, C', C^{\text{Mid}}$  for the circuit  $C'$  "halfway between"  $C^{\text{Left}}, C^{\text{Mid}}$ ; analogous for the right pair; if both surpass threshold,  $E$  repeats for both; and if neither surpass threshold, then  $E$  will not continue down this path of the binary search.

We prove that for appropriate choice of threshold,  $E$  will only ever visit polynomially many nodes in the binary search tree, and will be guaranteed to find a complete path for which  $\mathcal{A}$ 's distinguishing advantage maintains above threshold through all  $n$  steps down the tree (thus specifying a desired  $n$ -bit distinguishing input).

We now formalize this intuition.

*Proof of Theorem 6.2.* Fix a PPT adversary  $\mathcal{A}$  and polynomial  $p(k)$  corresponding to  $\mathcal{A}$ 's distinguishing advantage. For any pair of circuits  $C_0, C_1$  and every value  $v \in \{0, \dots, 2^n\}$ , define the circuit (taking  $n$ -bit inputs):

$$C_v(x) = \begin{cases} C_0(x) & \text{if } x < v \\ C_1(x) & \text{if } x \geq v \end{cases}.$$

Note that the size of  $C_v$  is  $O(|C_0| + |C_1|)$ . (We slightly abuse notation here, treating bit strings in some cases as integers and in others as bit strings; we will continue to do so for notational convenience).

Consider the following extractor algorithm  $E$  associated with  $\mathcal{A}$ , given a security parameter  $1^k$ , a pair of circuits  $C_0, C_1$ , and auxiliary input  $z$ . We assume  $E$  knows a polynomial upper bound  $d = d(k)$  on the number of differing points between  $C_0$  and  $C_1$ ; otherwise, it can simply perform a repeated doubling search, repeating the following algorithm for  $d(k) = k, k^2, k^{2^2}, k^{2^3}$ , etc. until successfully extracting (which is efficiently testable). For simplicity, we further assume that the adversary's original *signed* distinguishing advantage is positive (i.e., that  $\Pr[\mathcal{A}(1^k, \mathcal{O}(C_1), C_0, C_1, z)] > \Pr[\mathcal{A}(1^k, \mathcal{O}(C_0), C_0, C_1, z)]$ ); to treat both cases, the extraction algorithm can be run twice, once as written, and once with all signs and inequalities flipped.

Extractor  $E(1^k, C_0, C_1, z)$ :

1. Set  $\epsilon = 1/p(k)$ ,  $\text{err} = \epsilon/k^2d$  and  $\text{thresh} = \epsilon/kd$ . These values will not change during the recursion. Recall  $E$  must extract when  $\mathcal{A}$  has distinguishing advantage  $1/p(k)$ , and  $d = d(k)$  is an upper bound on the number of differing points of  $C_0, C_1$ .
2. Begin the recursion: Execute  $\text{Iterate}(n, \text{err}, \text{thresh}, v^L = 0, v^R = 2^n, v^M = 2^{n-1})$ , as defined in Figure 2.

`Iterate`( $m, \text{err}, \text{thresh}, v^L, v^R, v^M$ ):

1. If  $m = 1$  then BREAK (terminating all execution of  $E$ ) and RETURN  $v^L$  as the final extracted input value.
2. Estimate  $\mathcal{A}$ 's *signed* advantage  $\epsilon^L \in [-1, 1]$  in distinguishing between the distributions  $\{\mathcal{O}(C_{v^L})\}$  and  $\{\mathcal{O}(C_{v^M})\}$  with additional inputs  $1^k, C_{v^L}, C_{v^M}, z$ , using sampling such that the *additive* estimation error is greater than  $\text{err}$  with at least  $1 - 2^{-k}$  probability. (Note this requires  $O(k/\text{err}^2)$  samples).
3. Repeat the process to estimate  $\mathcal{A}$ 's *signed* advantage  $\epsilon^R$  in distinguishing between  $\{\mathcal{O}(C_{v^M})\}$  and  $\{\mathcal{O}(C_{v^R})\}$  within the same parameters.
4. Based on the values of  $\epsilon^L, \epsilon^R$ , iterate as follows:
  - If  $\epsilon^L > \text{thresh}$ : Continue along the left path in the binary search. That is, take  $v^R \leftarrow v^M$  and  $v^M \leftarrow \lceil (v^R - v^L)/2 \rceil$ , and recurse as `Iterate`( $m - 1, \text{err}, \text{thresh}, v^L, v^R, v^M$ ).
  - If  $\epsilon^R > \text{thresh}$ : Continue along the right path in the binary search. That is, take  $v^L \leftarrow v^M$  and  $v^M \leftarrow \lceil (v^R - v^L)/2 \rceil$ , and recurse as `Iterate`( $m - 1, \text{err}, \text{thresh}, v^L, v^R, v^M$ ).

Note that if both  $\epsilon^L, \epsilon^R$  surpass the threshold, then *both* paths of recursion are followed.

**Figure 2:** Recursion algorithm `Iterate`, used by the extractor  $E$ .

We now prove that, if  $\mathcal{A}$  distinguishes between obfuscations of a pair of circuits  $C_0, C_1$  with advantage  $1/p(k)$  then  $E$  successfully extracts an input  $x$  for which  $C_0(x) \neq C_1(x)$  with noticeable probability.

**Lemma 6.3.** *Suppose there exists a polynomial  $d(k)$ , circuits  $C_0, C_1 \in \mathcal{C}_k$  disagreeing on  $d(k)$  inputs, and auxiliary input  $z$  for which*

$$\Pr \left[ b \leftarrow \{0, 1\}; \tilde{C} \leftarrow \mathcal{O}(1^k, C_b) : \mathcal{A}(1^k, \tilde{C}, C_0, C_1, z) = b \right] \geq \frac{1}{2} + \frac{1}{p(k)}. \quad (3)$$

*Then the algorithm  $E$  on input  $(1^k, C_0, C_1, z)$  terminates within time  $t_E(k, d(k))$  for fixed polynomial  $t_E$ , and it holds that*

$$\Pr[v \leftarrow E(1^k, C_0, C_1, z) : C_0(v) \neq C_1(v)] \geq 1 - \text{negl}(k).$$

*Proof.* Denote by  $D \subset \{0, 1\}^n$  the subset of inputs on which  $C_0$  and  $C_1$  differ: i.e.,  $D = \{x \in \{0, 1\}^n : C_0(x) \neq C_1(x)\}$ , and  $d(k) = |D|$ . The lemma follows from three claims:

1. The runtime of  $E$  is bounded by  $t_E(k, d(k))$  for fixed polynomial  $t_E$ .
2. If  $E$  reaches `Iterate` at level  $m = 1$ , then it succeeds in extracting.
3. If the number of differing inputs of  $C_0, C_1$  is bounded by  $d(k)$ , and  $\mathcal{A}$ 's advantage in distinguishing  $\{\mathcal{O}(C_0)\}, \{\mathcal{O}(C_1)\}$  is at least  $1/p(k)$ , then with overwhelming probability  $E$  will reach an execution of `Iterate` at level  $m = 1$ .

**Claim 1.** The runtime of  $E$  is bounded by  $t_E(k, d(k))$  for some fixed polynomial  $t_E$ .

*Proof.* Associate the possible paths in the binary search process of  $E$  with a binary tree. Denote by  $T(m, \ell)$  an upper bound on the runtime of the iteration process when initiated on a node in the binary search tree at level  $m$  (i.e., containing  $2^m$  leaf nodes), and where  $\ell$  of its descendent leaf nodes correspond to inputs  $x \in D$ . In particular,  $T(n(k), d(k))$  is a bound on the total runtime of  $E$ . We make the following observations:

- $T(m, 0) = 0$  for any level  $m$ . Namely, any such path will yield a negligible distinguishing advantage (since all intermediate circuits will be equivalent), which for sufficiently large  $k$  will not exceed the threshold  $\epsilon/kd$ , and hence such a path will never be taken by  $E$ .
- $T(m, \ell) \leq q(k, d(k)) + T(m - 1, \ell^L) + T(m - 1, \ell^R)$ , where  $\ell^L, \ell^R$  denote the number of leaf nodes  $x \in D$  below the left and right child of the current node (so that  $\ell^L + \ell^R = \ell$ ), and  $q$  is a fixed polynomial.

Indeed, in each level of iteration,  $E$  runs  $O(k(1/\text{err}^2)) = O(k^5 d^2 / \epsilon^2)$  executions of the adversary algorithm  $\mathcal{A}$  (which runs in polynomial time  $t(k)$ ) in order to estimate  $\mathcal{A}$ 's distinguishing advantage to both the left and the right in the binary search tree, and then in the worst case recurses along both paths. Recalling  $\epsilon = 1/p(k)$ , we have  $q(k, d(k)) = O(t(k) \cdot k^5 d(k)^2 p(k)^2)$ .

Therefore, combining the two relations, it must be that  $T(n, d(k)) \leq q(k, d(k)) \cdot N$ , where  $N$  is equal to the total number of nodes in the binary search tree containing a nonzero number of leaf nodes  $x \in D$  beneath them. But, we know that  $N \leq n(k) \cdot d(k)$ , since there are  $d(k)$  relevant leaf nodes, and the depth of the tree is  $n$ . Hence, the claim holds.  $\square$

**Claim 2.** Suppose  $E$  returns a value  $v$ . Then it holds that  $C_0(v) \neq C_1(v)$ .

*Proof.* Recall that  $E$  returns a value  $v$  upon execution of `Iterate` at level  $m = 1$  for some triple of values  $v^L, v^R, v^M$ . In order for such a call to be made, it must be in the previous step that the measured distinguishing advantage of  $\mathcal{A}$  between obfuscations of the “left” and “right” circuits  $C_{v^L}$  and  $C_{v^R}$  surpassed the threshold value  $\text{thresh} = \epsilon/kd$ . By the choice of sampling parameters, together with a Chernoff bound, it then holds with overwhelming probability (since  $E$  visits only polynomially many nodes) that  $\mathcal{A}$ 's *true* advantage at this node is at least  $\text{thresh} - \text{err} = \epsilon/kd - \epsilon/k^2 d$ , which is a fixed noticeable function of  $k$  (recall  $\epsilon = 1/p(k)$ ). By the security of the indistinguishability obfuscator  $\mathcal{O}$ , it must be that  $C_{v^L} \neq C_{v^R}$ . But, since we are at level  $m = 1$ , the circuits  $C_{v^L}$  and  $C_{v^R}$  are *identical* except on input  $v^L$ , on which  $C_{v^L}(v^L) = C_0(v^L)$  and  $C_{v^R}(v^L) = C_1(v^L)$ . Thus, the returned value  $v^L$  must be a differing input, as desired.  $\square$

**Claim 3.** Suppose  $|D| \leq d(k)$  and  $\mathcal{A}$  distinguishes between obfuscations of  $C_0, C_1$  with *signed* advantage  $1/p(k)$ : i.e.,  $\Pr[\mathcal{A}(1^k, \mathcal{O}(C_1), C_0, C_1, z)] \geq \Pr[\mathcal{A}(1^k, \mathcal{O}(C_0), C_0, C_1, z)] + 1/p(k)$ . Then with overwhelming probability,  $E$  will return some value  $v$ .<sup>11</sup>

*Proof.* Let  $\alpha \geq 1/p(k)$  denote the true signed distinguishing advantage of  $\mathcal{A}$  between the obfuscations  $\{\mathcal{O}(C_0)\}$  and  $\{\mathcal{O}(C_1)\}$ . For each node in the binary search tree, associate with this node the true signed distinguishing advantage of  $\mathcal{A}$  for the corresponding pair of obfuscated circuits (maintaining canonical fixed ordering for the sign). For example, the root node corresponds to value  $\alpha$ , its left child corresponds to the signed probability difference that  $\mathcal{A}$  outputs 1 given an obfuscation of  $C_0$  and the “middle” circuit  $C_{2^{n-1}}$ , and so on.

Consider a node  $v'$ , its labelled distinguishing advantage  $\epsilon'$ , and the number of its descendent leaf nodes  $d'$  contained in  $D$ . By definition, for any node  $v'$  in the tree and its two children it holds that  $\epsilon^L + \epsilon^R = \epsilon'$  (this is true as we consider *signed* advantage values) and  $d^L + d^R = d'$ . We consider two different cases for how these values split from  $v'$  among its children.

<sup>11</sup>Recall to address the case of *negative* signed advantage  $-1/p(k)$ , the extraction algorithm  $E$  can be run a second time, with all signs and inequalities flipped.

- Claim 3.1: If  $d' = d^b > 0$  for  $b \in \{L, R\}$ : Then  $\epsilon^b > \epsilon' - \mu(k)$  for negligible function  $\mu(k)$ .  
Note that the condition  $d' = d^b$  means that  $d^{\bar{b}} = 0$  for the other path  $\bar{b} \neq b$ . As argued above, by the security of the indistinguishability obfuscator, for each node in the tree which does *not* have any descendent leaf nodes  $x \in D$  corresponding to a differing input, this node must correspond to some negligible distinguishing advantage  $\epsilon^b < \mu(k)$ . Thus, we have  $\epsilon^b > \epsilon' - \mu(k)$ .
- Claim 3.2: If  $d' > d^L, d^R$ : Then there exists  $b \in \{L, R\}$  for which  $\epsilon^b/d^b \geq \epsilon'/d'$ .  
Suppose to the contrary we have  $\epsilon^b/d^b < \epsilon'/d'$  for both values of  $b \in \{L, R\}$ . That is,  $\epsilon^b d' < \epsilon' d^b$ . Then, combining these expressions for both values of  $b$ , it must be that

$$(\epsilon^L + \epsilon^R)d' < \epsilon'(d^L + d^R). \quad (4)$$

But we have that  $\epsilon^L + \epsilon^R = \epsilon'$  and  $d^L + d^R = d'$ , and so Inequality (4) yields the contradiction  $\epsilon' d' < \epsilon' d'$ . Thus, the claim holds.

Note that we are not concerned with the third case of  $d' = 0$ , as we are interested only in the paths down to the differing inputs  $x \in D$  (which have  $d' > 0$  at all intermediate nodes).

Combining the two sub-claims, we have that for *every* node with  $d' > 0$ , there must exist  $b \in \{L, R\}$  for which  $d^b \neq 0$  and  $\epsilon^b/d^b \geq \epsilon'/d' - \mu(k)$ . Indeed, this follows directly for the case of Claim 3.2, and in the case of Claim 3.1 it holds since  $\epsilon^b \geq \epsilon' - \mu(k)$  and  $d^b = d' \geq 1$  for some  $b \in \{L, R\}$ .

Thus, since we begin at the root node with ratio  $\alpha/d$  (corresponding to  $\mathcal{A}$ 's original signed distinguishing advantage  $\alpha$  and the total number of differing inputs  $d = d(k)$ ), then by applying the above sub-claims  $n$  times inductively, it must be the case that for some complete path down the tree, *every* intermediate node along the path satisfies

$$\frac{\epsilon'}{d'} \geq \frac{\alpha}{d} - n \cdot \mu(k) \geq \frac{\epsilon}{2d},$$

where the second inequality holds for sufficiently large  $k$ , since  $\mu(k)$  is negligible in  $k$ , and we have  $\alpha \geq \epsilon$ . In particular, since  $d' \geq 1$ , it must be that  $\epsilon' \geq \epsilon/2d$  for each node along the path.

Now, consider the algorithm  $E$ . Recall that  $E$  estimates the distinguishing advantage of  $\mathcal{A}$  at each visited node via sampling with additive estimation error  $\text{err} = \epsilon/k^2 d$ , and continues down a path if the measured value surpasses  $\text{thresh} = \epsilon/kd$ . By a Chernoff bound (since  $E$  only visits polynomially many nodes), it holds with overwhelming probability that  $E$  successfully estimates the true distinguishing probability label of each visited node within additive error  $\text{err} = \epsilon/k^2 d$ . But then, for each node along the path above, the measured value must be at least  $\epsilon/2d - \text{err} = \epsilon/2d - \epsilon/k^2 d \geq (\epsilon/d)(1/2 - 1/k^2) \geq \epsilon/kd = \text{thresh}$ . Therefore,  $E$  will necessarily follow this path in the binary search path down all the way to the associated leaf  $x^*$ . The claim follows. □

Combining Claims 1-3, this concludes the proof of Lemma 6.3. □

Therefore, by the existence of the extractor algorithm  $E$  constructed above, it follows that the indistinguishability obfuscator  $\mathcal{O}$  is also a weak extractability obfuscator. □

Note that Theorem 6.2 implies, for example, that for the class of polynomial multipoint locker functions (i.e., functions evaluating to nonzero bit strings at polynomially many hidden points), indistinguishability obfuscation is *equivalent* to extractability obfuscation.

## 6.2 From Indistinguishability FWE to FWE for Languages with Polynomial Witnesses

We now address this question in the language of FWE.

Mirroring the definition of indistinguishability obfuscation, we define a weaker notion of FWE—which we refer to as *indistinguishability FWE*—which only requires that if  $f(m_0, w) = f(m_1, w)$  for *all* witnesses  $w$  for  $x \in L$ , then encryptions of  $m_0$  and  $m_1$  are indistinguishable. Recall that, in contrast, the stronger notion requires that if you can distinguish between encryptions of  $m_0$  and  $m_1$  you must know a witness on which they differ.

**Definition 6.4** (Indistinguishability Functional Witness Encryption). An *indistinguishability functional witness encryption* (iFWE) scheme for an NP language  $L$  (with corresponding relation  $R$ ) and class of functions  $\mathcal{F} = \{F_k\}$  consists of encryption and decryption algorithms  $\text{Enc}, \text{Dec}$  with the same syntax as standard FWE, satisfying the same correctness property, and the following (weaker) security property:

**(Indistinguishability) security:** For every PPT adversary  $\mathcal{A}$  and polynomial  $\ell(\cdot)$ , there exists a negligible function  $\nu(\cdot)$  such that for every security parameter  $k$ , every function  $f \in F_k$ , pair of messages  $m_0, m_1 \in \text{MSG}_k$ , a string  $x$ , and an auxiliary information  $z$  of length at most  $\ell(k)$  for which  $f(m_0, w) = f(m_1, w)$  for *every* witness  $w$  of  $x \in L$ ,

$$\left| \Pr \left[ \mathcal{A}(1^k, \text{Enc}(1^k, x, m_0, f), z) = 1 \right] - \Pr \left[ \mathcal{A}(1^k, \text{Enc}(1^k, x, m_1, f), z) = 1 \right] \right| < \nu(k).$$

Using the same transformation as in the Extractability Obfuscation-FWE equivalence (see Theorem 4.3), it can be seen that indistinguishability FWE for  $P/\text{poly}$  and  $NP$  is directly equivalent to indistinguishability obfuscation for  $P/\text{poly}$ .

**Theorem 6.5** (Equivalence of Indistinguishability FWE and Indistinguishability Obfuscation). *The existence of the following two primitives are equivalent:*

- *Succinct indistinguishability functional witness encryption for  $NP$  and  $P/\text{poly}$ .*
- *Succinct indistinguishability obfuscation for  $P/\text{poly}$ .*

We now consider the question of whether we can turn any indistinguishability FWE into an FWE. We provide an affirmative answer for two restricted cases.

The first result is derived from the indistinguishability obfuscation to weak extractability obfuscation transformation from the previous section. Loosely, it says that from indistinguishability FWE for  $P/\text{poly}$ , we can obtain a weak form of FWE where (extraction) security holds as long as the function  $f(m, w)$  is not “too sensitive” to  $m$ : i.e., if for any two messages  $m_0, m_1$  there are only polynomially many witnesses  $w$  for which  $f(m_0, w) \neq f(m_1, w)$ . For example, this captures functions  $f$  that only rarely output nonzero values. Going back to the example of encrypting data  $m$  associated with nodes of a social network, we could then allow someone holding clique  $w$  to learn whether the nodes in this clique satisfy some chosen rare property (e.g., contains someone with a rare disease, all have the same birthday, etc). Then, while there may be many cliques (corresponding to several, even exponentially many, witnesses  $w$ ), it will hold that  $f(m, w)$  is almost always 0, for all but polynomially many  $w$ .

The second result considers indistinguishability FWE for general function classes (instead of just  $P/\text{poly}$ ), but restricts to  $NP$  languages with polynomial witnesses. In the encrypted social network example, this allows basing on a weaker assumption (not requiring the original iFWE scheme to support all  $P/\text{poly}$ ), but would restrict to social networks with only polynomially many

cliques. The transformation preserves the supported function class: For example, given iFWE for the singleton function class  $\{f(m, w) = m\}$  (corresponding to standard witness encryption), one obtains standard FWE for the same class (i.e., *extractable* witness encryption). This result requires a new approach, and makes use of techniques in error-correcting codes.

We proceed to elaborate our first result. First, consider the implications of the transformation from the previous section within the setting of FWE. Suppose we start with an iFWE scheme for  $NP$  and  $P/\text{poly}$ . Recall that such a scheme is equivalent to an indistinguishability obfuscator for  $P/\text{poly}$ , up to a simple transformation. Then using Theorem 6.2, from this we obtain a *weak* extractability obfuscator (as per Definition 6.1).

Now, consider the result of applying the simple extractability obfuscation-to-FWE equivalence transformation (see Theorem 4.3) to this weak extractability obfuscator. Recall that in the transformation, a message  $m$  is encrypted with respect to  $NP$  relation  $R$ , instance  $x$  and function  $f$  by obfuscating the function  $f'_m$  that on input  $w$  verifies whether  $R(x, w) = 1$  and, if so, outputs  $f(m, w)$ . The obfuscated program constitutes the ciphertext of  $m$ . If the obfuscator is a standard extractability obfuscator, then the resulting scheme is a standard FWE. However, starting with a *weak* extractability obfuscator (which only guarantees extraction for function pairs  $f_0, f_1$  that agree on all but polynomially many inputs), then we will only achieve extraction in the resulting FWE in the case that the obfuscated programs  $f'_m$  for different messages  $m$  agree on all but polynomially many inputs. That is, when  $R, x$ , and  $f$  satisfy  $f(m_0, w) = f(m_1, w)$  for all but polynomially many witnesses  $w$  of  $R(x, w) = 1$ , for any pair of messages  $m_0, m_1$ .

**Definition 6.6.** We say a class of functions  $\mathcal{F} = \{F_k\}$  has *t-bounded sensitivity* with respect to message space  $MSG$  and NP language  $L$  (with relation  $R$ ), if for every  $f \in F_k$ , every  $m_0, m_1 \in MSG$ , and every  $x \in \{0, 1\}^*$  there are at most  $t(|x|)$  witnesses  $w$  such that  $R(x, w) = 1$  and it holds that  $f(m_0, w) \neq f(m_1, w)$ .

As a special case, if the language  $L$  has only polynomially many witnesses for each statement, then this property is satisfied for any class of functions.

Putting the pieces together, we have the following corollary to Theorem 6.2.

**Corollary 6.7.** *Suppose there exists indistinguishability functional witness encryption for  $NP$  and  $P/\text{poly}$ . Then for any polynomial  $t(\cdot)$ , there exist functional witness encryption schemes for any class of functions  $\mathcal{F} = \{F_k\}$ , message space  $MSG$ , and NP language  $L$ , for which  $\mathcal{F}$  has *t-bounded sensitivity with respect to  $MSG$  and  $L$* .*

However, as discussed above, this result requires one to begin with indistinguishability FWE supporting all functions in  $P/\text{poly}$ . But what about the case of restricted function classes? For example, Corollary 6.7 provides no implications to the case of (standard) witness encryption and extractable witness encryption. We next provide a (slightly weaker) transformation for general function classes, for the special case of  $NP$  languages with polynomially many witnesses.

**Definition 6.8.** Let  $L$  be an NP language with corresponding relation  $R$ . We say that  $L$  has *t-bounded witness* if for every  $x \in \{0, 1\}^*$ , there are at most  $t(|x|)$  distinct witnesses  $w$  such that  $R(x, w) = 1$ .

**Theorem 6.9.** *For every function class  $\mathcal{F} = \{F_k\}$  and polynomial  $t(\cdot)$ , if there exist indistinguishability functional witness encryption schemes for  $\mathcal{F}$  and every *t-bounded witness* NP language, then for every *t-bounded witness* NP language  $L$  (with corresponding relation  $R$ ), there exists a functional witness encryption schemes for  $\mathcal{F}$  and  $L$ .*

*Proof.* Let  $L$  be a  $t$ -bounded witness NP language with corresponding relation  $R$  for some polynomial  $t(\cdot)$ . Define  $q(\cdot)$  such that for every  $k \in \mathbb{N}$ ,  $q(k)$  is the smallest prime  $\geq 8t(k)$ . Assume without loss of generality (by padding) that any witness of any  $x \in L$  has length  $u(|x|)$  for some polynomial  $u$ . To construct a functional witness encryption scheme  $(\text{Enc}, \text{Dec})$  for  $L$  and  $\mathcal{F}$ , we consider the following NP language  $L'$ .

$$L' = \{(x, r, a) : \exists w \in \{0, 1\}^{u(|x|)} \text{ s.t. } (R(x, w) = 1) \wedge (r \in \mathbb{F}_{q(|x|)}^{u(|x|)}) \wedge (\langle r, w \rangle = a)\},$$

where  $\mathbb{F}_q = \{0, \dots, q-1\}$  is the prime field of size  $q$  and  $\langle \cdot, \cdot \rangle$  denotes inner product over  $\mathbb{F}_q^u$ .

Let  $(\text{Enc}', \text{Dec}')$  be a indistinguishability functional witness encryption scheme for  $L'$  and  $\mathcal{F}$ . We construct a functional witness encryption scheme  $(\text{Enc}, \text{Dec})$  for  $L$  and  $\mathcal{F}$  as follows.

- $\text{Enc}(1^k, x, m, f)$ : On input the security parameter  $1^k$ , statement  $x \in \{0, 1\}^*$ , message  $m \in \text{MSG}_k$ , and function  $f \in \mathcal{F}_k$ ,  $\text{Enc}$  generates a ciphertext  $c$  as follows.
  - Let  $q = q(|x|)$  and  $u = u(|x|)$ . Sample  $r \leftarrow \mathbb{F}_q^u$  uniformly at random.
  - For every  $a \in \mathbb{F}_q$ , compute  $c_a = \text{Enc}'(1^k, (x, q, r, a), m, f)$ .
  - Output  $c = \{c_a\}_{a \in \mathbb{F}_q}$ .
- $\text{Dec}(c, w)$ : On input a ciphertext  $c = \{c_a\}_{a \in \mathbb{F}_q}$  and a witness  $w \in \{0, 1\}^*$ ,  $\text{Dec}$  runs  $\text{Dec}'(c_a, w)$  for every  $a \in \mathbb{F}_q$ . If there exists some  $a$  such that  $\text{Dec}'(c_a, w) \neq \perp$ , then output the first non- $\perp$   $\text{Dec}'(c_a, w)$ . Otherwise, output  $\perp$ .

It is not hard to see that correctness of  $(\text{Enc}', \text{Dec}')$  implies correctness of  $(\text{Enc}, \text{Dec})$ : For every  $k, x, m, f, w$ , if  $w$  is a witness for  $x \in L$ , then there exists some  $a \in \mathbb{F}_q$  such that  $w$  is a witness for  $(x, q, r, a) \in L'$ , and for the first such  $a$ , by the correctness of  $(\text{Enc}', \text{Dec}')$ ,  $\text{Dec}'(c_a, w) = f(m, w)$  with  $1 - \text{negl}(k)$  probability, which implies that  $\text{Dec}(\text{Enc}(1^k, x, m, f), w)$  output  $f(m, w)$  with  $1 - \text{negl}(k)$  probability as well.

We proceed to prove security of  $(\text{Enc}, \text{Dec})$ . At a high level, we show that if an adversary  $\mathcal{A}$  can distinguish  $\text{Enc}(1^k, x, m_0, f)$  and  $\text{Enc}(1^k, x, m_1, f)$  with a non-negligible advantage, then there is a non-negligible fraction of  $r \in \mathbb{F}_q^u$  such that we learn non-trivial information about the value of  $\langle r, w \rangle$  for some witness  $w$  such that  $f(m_0, w) \neq f(m_1, w)$ . Note that a linear function  $g_w(r) := \langle r, w \rangle$  can be viewed as a  $q$ -ary Hadamard code of  $w$ . The non-trivial information allows us to obtain a (randomized) function  $h(r)$  that agree with  $g_w(r)$  on non-negligibly more than  $1/q$  fraction of points. We can then apply the local list-decoding algorithm of Goldreich, Rubinfeld, and Sudan [GRS00] to recover  $w$ .

Formally, let  $\mathcal{A}$  be a PPT adversary and  $\ell$  be a polynomial. We construct a PPT extractor  $E$  for  $\mathcal{A}$  as follows. Let  $\tilde{q}$  be a polynomial, and fix a security parameter  $k$ . Given two messages  $m_0, m_1 \in \text{MSG}_k$ , a function  $f \in \mathcal{F}$ , a string  $x$  and an auxiliary information  $z$  of length at most  $\ell$  such that

$$\Pr \left[ b \leftarrow \{0, 1\}; c \leftarrow \text{Enc}(1^k, x, m_b, f) : \mathcal{A}(1^k, c, z) = b \right] \geq \frac{1}{2} + \frac{1}{\tilde{q}(k)},$$

we first construct a randomized function  $h : \mathbb{F}_q^{u(|x|)} \rightarrow \mathbb{F}_q$  with non-trivial agreement to a  $q$ -ary Hadamard codeword  $g_w$  for some witness  $w$ .

For notational convenience, let  $c(m, r, a) = \text{Enc}'(1^k, (x, r, a), m, f)$ , and note that  $\text{Enc}(1^k, x, m_b, M) = \{c(m_b, r, a)\}_{a \in \mathbb{F}_q}$  for uniformly random  $r$ . By a standard averaging argument, it follows that with probability at least  $\epsilon = 1/2\tilde{q}(k)$  over  $r \leftarrow \mathbb{F}_q^u$ ,

$$\left| \Pr[\mathcal{A}(1^k, \{c(m_0, r, a)\}_{a \in \mathbb{F}_q}, z) = 1] - \Pr[\mathcal{A}(1^k, \{c(m_1, r, a)\}_{a \in \mathbb{F}_q}, z) = 1] \right| \geq \epsilon. \quad (5)$$

Now, for every  $r \in \mathcal{F}_q^u$  and  $a \in \{0, \dots, q\}$ , define hybrid distributions

$$\mathcal{D}_{r,a} = (c(m_1, r, 0), \dots, c(m_1, r, a-1), c(m_1, r, a), \dots, c(m_1, r, q-1)).$$

If  $r$  is such that Eq. (5) holds, then by an averaging argument, there exists some  $a^* \in [q]$  such that

$$|\Pr[\mathcal{A}(1^k, \mathcal{D}_{r,a^*-1}, z) = 1] - \Pr[\mathcal{A}(1^k, \mathcal{D}_{r,a^*}, z) = 1]| \geq \epsilon/q.$$

Note that  $\mathcal{D}_{r,a^*-1}$  and  $\mathcal{D}_{r,a^*}$  only differ in their  $a$ -th coordinate, which is  $c(m_0, r, a^*) = \text{Enc}'(1^k, (x, r, a^*), m_0, f)$  for  $\mathcal{D}_{r,a^*-1}$  and  $c(m_1, r, a^*) = \text{Enc}'(1^k, (x, r, a^*), m_1, f)$  for  $\mathcal{D}_{r,a^*}$ . Thus, (indistinguishability) security of  $(\text{Enc}', \text{Dec}')$  implies that there exists some  $w$  for  $x \in L$  such that (i)  $\langle w, r \rangle = a^*$  and (ii)  $f(m_0, w) \neq f(m_1, w)$ . Furthermore, let  $S_r \subseteq \mathbb{F}_q$  be the set of  $a \in \mathbb{F}_q$  such that  $|\Pr[\mathcal{A}(1^k, \mathcal{D}_{r,a-1}, z) = 1] - \Pr[\mathcal{A}(1^k, \mathcal{D}_{r,a}, z) = 1]| \geq \epsilon/4q$ . (Indistinguishability) security of  $(\text{Enc}', \text{Dec}')$  implies that for every  $a \in S_r$ , there exists some  $w$  for  $x \in L$  such that (i)  $\langle w, r \rangle = a$  and (ii)  $f(m_0, w) \neq f(m_1, w)$ . Recall that there are at most  $t$  witnesses for  $x \in L$ , so  $|S_r| \leq t$ .

Now, let us consider the following (efficient) randomized function  $h : \mathbb{F}_q^u \rightarrow \mathbb{F}_q$ . On input  $r \in \mathbb{F}_q^u$ ,  $h$  performs the following.

- For each  $a \in \mathbb{F}_q$ ,  $h$  estimates the distinguishing gap  $|\Pr[\mathcal{A}(1^k, \mathcal{D}_{r,a-1}, z) = 1] - \Pr[\mathcal{A}(1^k, \mathcal{D}_{r,a}, z) = 1]|$  using sampling such that the estimation error is  $\leq \epsilon/10q$  with at least  $1 - 2^{-k}$  probability.
- Let  $S'_r$  be the subset of  $\mathbb{F}_q$  such that the estimated distinguishing gap is  $\geq \epsilon/3q$ . If  $S'_r$  is non-empty, then  $h$  outputs a random element in  $S'_r$ . Otherwise,  $h$  outputs a random element in  $\mathbb{F}_q$ .

**Claim 6.10.** *There exists a witness  $w$  for  $x \in L$  such that  $f(m_0, w) \neq f(m_1, w)$  and*

$$\Pr[r \leftarrow \mathbb{F}_q^u : h(r) = \langle w, r \rangle] \geq 1/q + \epsilon/2t.$$

*Proof.* For every  $r \in \mathbb{F}_q^u$ , by an union bound, with probability at least  $1 - q \cdot 2^{-k}$  all estimation has error at most  $\epsilon/10q$ . In this case,  $S'_r \subset S_r$  and if  $r$  satisfies Eq. (5), then  $S'_r$  is non-empty (at least contains  $a^*$ ). Let  $t' \leq t$  be the number of witnesses for  $x \in L$  such that  $f(m_0, w) \neq f(m_1, w)$ . Recall that at least  $\epsilon$ -fraction of  $r$  satisfies Eq. (5). We have

$$\Pr[r \leftarrow \mathbb{F}_q^u, a \leftarrow h(r) : \exists w \text{ for } x \in L \text{ s.t. } \langle w, r \rangle = a] \geq \epsilon + (1 - \epsilon) \cdot t'/q - q \cdot 2^{-k}.$$

Therefore, by averaging, there exists a witness  $w$  such that

$$\Pr[r \leftarrow \mathbb{F}_q^u : h(r) = \langle w, r \rangle] \geq (\epsilon + (1 - \epsilon) \cdot t'/q - q \cdot 2^{-k})/t' \geq 1/q + \epsilon/2t.$$

□

The above claim says that  $h$  has non-trivial (i.e., non-negligibly greater than  $1/q$ ) agreement with the  $q$ -ary Hadamard code  $g_w$  of a desired witness  $w$ . Note that by a standard Chernoff bound, with overwhelming probability over the randomness of  $h$  (where each input  $r$  use independent coins),  $h$  with fixed randomness has at least  $1/q + \epsilon/4t$  agreement with  $g_w$ . We can then finish the proof by letting the extractor  $E$  invoke the local list-decoding algorithm of Goldreich, Rubinfeld, and Sudan [GRS00] from the following theorem to recover  $w$  in polynomial time.

**Theorem 6.11** ([GRS00]). *Let  $\mathbb{F}_q$  be a prime field,  $h : \mathbb{F}_q^u \rightarrow \mathbb{F}_q$  be a function. Let  $k \in \mathbb{N}$  and  $\epsilon \in (0, 1)$  be parameters. There exists an algorithm that given  $k, \epsilon$  and oracle access to  $h$ , runs in  $\text{poly}(ku/\epsilon)$  time and with probability at least  $1 - 2^{-k}$  outputs a list of linear polynomials that contains all linear polynomials that agree with  $h$  on at least  $1/q + \epsilon$  fraction of points.*



□

We remark that, in particular, Theorem 6.9 gives a method for transforming indistinguishability FWE for the trivial function  $f(m, w) = m$  to FWE for the same function  $f$ . It is easy to see that indistinguishability FWE for this particular  $f$  is equivalent to the notion of witness encryption [?], and FWE for the same  $f$  is equivalent to the notion of extractable witness encryption of [GKP<sup>+</sup>13]. Theorem 6.9 thus shows how to turn witness encryption to extractable witness encryption for the case of languages with polynomially many witness.

## References

- [BCCT13] Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. Recursive composition and bootstrapping for snarks and proof-carrying data. In *STOC*, pages 111–120, 2013.
- [BCI<sup>+</sup>13] Nir Bitansky, Alessandro Chiesa, Yuval Ishai, Rafail Ostrovsky, and Omer Paneth. Succinct non-interactive arguments via linear interactive proofs. In *TCC*, pages 315–333, 2013.
- [BCPR13] Nir Bitansky, Ran Canetti, Omer Paneth, and Alon Rosen. Indistinguishability obfuscation vs. auxiliary-input extractable functions: One must fall. Cryptology ePrint Archive, Report 2013/641, 2013.
- [BF13] Mihir Bellare and Georg Fuchsbauer. Policy-based signatures. Cryptology ePrint Archive, Report 2013/413, 2013.
- [BGI<sup>+</sup>12] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. *J. ACM*, 59(2):6, 2012.
- [BGI13] Elette Boyle, Shafi Goldwasser, and Ioana Ivan. Functional signatures and pseudorandom functions. Cryptology ePrint Archive, Report 2013/401, 2013.
- [BGK<sup>+</sup>13] Boaz Barak, Sanjam Garg, Yael Tauman Kalai, Omer Paneth, and Amit Sahai. Protecting obfuscation against algebraic attacks. Cryptology ePrint Archive, Report 2013/631, 2013.
- [BGV11] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. Fully homomorphic encryption without bootstrapping. *Electronic Colloquium on Computational Complexity (ECCC)*, 18:111, 2011.
- [BP13] Elette Boyle and Rafael Pass. Limits of extractability assumptions with distributional auxiliary input. Cryptology ePrint Archive, Report 2013/703, 2013.
- [BR13] Zvika Brakerski and Guy N. Rothblum. Virtual black-box obfuscation for all circuits via generic graded encoding. Cryptology ePrint Archive, Report 2013/563, 2013.
- [BSW12] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: a new vision for public-key cryptography. *Commun. ACM*, 55(11):56–64, 2012.
- [BV11] Z. Brakerski and V. Vaikuntanathan. Efficient fully homomorphic encryption from (standard) lwe. To appear in TCC 2010, 2011.
- [BV13] Zvika Brakerski and Vinod Vaikuntanathan. Lattice-based fhe as secure as pke. Cryptology ePrint Archive, Report 2013/541, 2013.

- [CLP12] Kai-Min Chung, Huijia Lin, and Rafael Pass. Constant-round concurrent zero knowledge from falsifiable assumptions. Cryptology ePrint Archive, Report 2012/563, 2012.
- [CLP13] Ran Canetti, Huijia Lin, and Omer Paneth. Public-coin concurrent zero-knowledge in the global hash model. In *TCC*, pages 80–99, 2013.
- [Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *STOC*, pages 169–178, 2009.
- [GGH<sup>+</sup>13] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *FOCS*, 2013.
- [GGHW13] Sanjam Garg, Craig Gentry, Shai Halevi, and Daniel Wichs. On the implausibility of differing-inputs obfuscation and extractable witness encryption with auxiliary input. Cryptology ePrint Archive, Report 2013/860, 2013.
- [GGPR13] Rosario Gennaro, Craig Gentry, Bryan Parno, and Mariana Raykova. Quadratic span programs and succinct nizks without pcs. In *EUROCRYPT*, pages 626–645, 2013.
- [GGSW13] Sanjam Garg, Craig Gentry, Amit Sahai, and Brent Waters. Witness encryption and its applications. In *STOC*, pages 467–476, 2013.
- [GKP<sup>+</sup>13] Shafi Goldwasser, Yael Tauman Kalai, Raluca A. Popa, Vinod Vaikuntanathan, and Nikolai Zeldovich. How to run turing machines on encrypted data. In *CRYPTO (2)*, pages 536–553, 2013.
- [Gro10] Jens Groth. Short pairing-based non-interactive zero-knowledge arguments. In *ASIACRYPT*, pages 321–340, 2010.
- [GRS00] Oded Goldreich, Ronitt Rubinfeld, and Madhu Sudan. Learning polynomials with queries: The highly noisy case. *SIAM J. Discrete Math.*, 13(4):535–570, 2000.
- [GSW13] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *CRYPTO (1)*, pages 75–92, 2013.
- [Had00] Satoshi Hada. Zero-knowledge and code obfuscation. In *ASIACRYPT*, pages 443–457, 2000.
- [HL02] Jeremy Horwitz and Ben Lynn. Toward hierarchical identity-based encryption. In *EUROCRYPT*, pages 466–481, 2002.
- [HSW13] Susan Hohenberger, Amit Sahai, and Brent Waters. Replacing a random oracle: Full domain hash from indistinguishability obfuscation. Cryptology ePrint Archive, Report 2013/509, 2013. <http://eprint.iacr.org/>.
- [HT98] Satoshi Hada and Toshiaki Tanaka. On the existence of 3-round zero-knowledge protocols. In Hugo Krawczyk, editor, *CRYPTO*, volume 1462 of *Lecture Notes in Computer Science*, pages 408–423. Springer, 1998.
- [Lip12] Helger Lipmaa. Progression-free sets and sublinear pairing-based non-interactive zero-knowledge arguments. In *TCC*, pages 169–189, 2012.
- [Mic00] Silvio Micali. Computationally sound proofs. *SIAM J. Comput.*, 30(4):1253–1298, 2000.
- [Nao03] Moni Naor. On cryptographic assumptions and challenges. In *CRYPTO*, pages 96–109, 2003.

- [O’N10] Adam O’Neill. Definitional issues in functional encryption. Cryptology ePrint Archive, Report 2010/556, 2010. <http://eprint.iacr.org/>.
- [SW13] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: Deniable encryption, and more. Cryptology ePrint Archive, Report 2013/454, 2013. <http://eprint.iacr.org/>.

## A Impossibility of Black-Box Extractability Definition

One may consider a stronger notion of extractability obfuscation, where the extractor algorithm  $E$  is not given access to the circuits  $C_0, C_1$  on which  $\mathcal{A}$  distinguishes (and on which  $E$  must extract a disagreeing input), but rather is only allowed *black-box* access to the two circuits in question. That is, for every PPT adversary  $\mathcal{A}$  and polynomial  $p(k)$ , this definition would require the existence of a PPT extractor algorithm  $E$  and polynomial  $q(k)$  such that for every security parameter  $k$ , every auxiliary input  $z$ , and every pair of circuits  $C_0, C_1 \in \mathcal{C}_k$ , if

$$\left| \Pr[\tilde{C} \leftarrow \mathcal{O}(C_0) : 1 \leftarrow \mathcal{A}(1^k, z, C_0, C_1, \tilde{C})] - \Pr[\tilde{C} \leftarrow \mathcal{O}(C_1) : 1 \leftarrow \mathcal{A}(1^k, z, C_0, C_1, \tilde{C})] \right| \geq \frac{1}{2} + \frac{1}{p(k)},$$

then

$$\Pr[x \leftarrow E^{C_0(\cdot), C_1(\cdot)}(1^k, z) : C_0(x) \neq C_1(x)] \geq \frac{1}{q(k)}.$$

As we now show, however, such a definition is impossible to achieve.

To demonstrate impossibility, we draw upon a class of circuits that was shown to be unobfuscatable in the virtual black box setting [?]. Namely, let  $(\text{Gen}, \text{Enc}, \text{Dec}, \text{Eval})$  be a semantically secure fully homomorphic encryption scheme for  $n$ -bit messages and ciphertext size  $N$ . For each security parameter  $k$ , consider the class of circuits

$$\mathcal{C}_k = \{C_{k,a,b,v,\text{pk},\text{sk},\hat{a}}\}_{a,b,v \in \{0,1\}^k, (\text{pk},\text{sk}) \in \text{Gen}(1^k), \hat{a} \in \text{Enc}(\text{pk},a)},$$

taking  $N$ -bit inputs, where

$$C_{k,a,b,v,\text{pk},\text{sk},\hat{a}}(x) = \begin{cases} (\text{pk}, \hat{a}) & \text{if } x = 0 \\ b & \text{if } x = a \\ v & \text{if } \text{Dec}(\text{sk}, x) = b \\ 0 & \text{else} \end{cases}.$$

Note that given any circuit  $\tilde{C}$  evaluating  $C_{k,a,b,v,\text{pk},\text{sk},\hat{a}}$ , one can *homomorphically* evaluate  $\tilde{C}$  on the received ciphertext  $\hat{a}$  (given “for free” by  $\tilde{C}$ ), in order to generate a valid encryption of the hidden value  $b$ , and then can feed this new ciphertext back into  $\tilde{C}$  to reveal the secret string  $v$ . Thus, there exists a PPT algorithm  $\mathcal{A}$  which, given any obfuscation  $\tilde{C} \leftarrow \mathcal{O}(C_{k,a,b,v,\text{pk},\text{sk},\hat{a}})$  (and auxiliary input  $z = \emptyset$ ), succeeds with probability 1 in identifying the vector  $v$ . In particular, for every  $a, b, v, \text{pk}, \text{sk}, \hat{a}$ ,  $\mathcal{A}$  will distinguish between obfuscations  $\{\mathcal{O}(C_{k,a,b,v,\text{pk},\text{sk},\hat{a}})\}$  and  $\{\mathcal{O}(C_{k,a,b,0,\text{pk},\text{sk},\hat{a}})\}$ .

Now, since  $\mathcal{O}$  is assumed to satisfy the above BB-extractability obfuscation security, there must exist a PPT extractor algorithm  $E$  and a polynomial  $q(k)$  such that, for every  $a, b, v, \text{pk}, \text{sk}, \hat{a}$ , then given only *black-box* access to the circuits  $C_{k,a,b,v,\text{pk},\text{sk},\hat{a}}$  and  $C_{k,a,b,0,\text{pk},\text{sk},\hat{a}}$ , the algorithm  $E$  extracts an input  $x$  for which  $C_{k,a,b,v,\text{pk},\text{sk},\hat{a}}(x) \neq C_{k,a,b,0,\text{pk},\text{sk},\hat{a}}(x)$  with probability  $1/q(k)$ . In particular, for

empty auxiliary input  $z = \emptyset$ , it holds that

$$\Pr \left[ (\text{pk}, \text{sk}) \leftarrow \text{Gen}(1^k); a, b, v \leftarrow \{0, 1\}^n; \hat{a} \leftarrow \text{Enc}(\text{pk}, a); x \leftarrow E^{C_0(\cdot), C_1(\cdot)}(1^k, z) : C_0(x) \neq C_1(x) \right] \geq \frac{1}{q(k)},$$

where  $C_0 := C_{k,a,b,v,\text{pk},\text{sk},\hat{a}}$  and  $C_1 := C_{k,a,b,0,\text{pk},\text{sk},\hat{a}}$ .

However, we now argue that such an extractor violates the semantic security of the FHE scheme. Recall the circuits  $C_0, C_1$  differ only on inputs  $x$  for which  $\text{Dec}(\text{sk}, x) = b$  (on these inputs  $C_0(x) = v$ , whereas  $C_1(x) = 0$ ). Since  $b$  was randomly chosen from an exponentially large set of values, to find such an input with noticeable probability, the extractor *must* query one of the circuits on input  $a$ , otherwise his view is independent of  $b$ . But, if the original ciphertext  $c$  is an encryption of 0 instead of  $a$ , then the view of  $E$  is also independent of  $a$ , and thus this cannot occur.

More formally, consider the following adversary  $\mathcal{A}_{\text{FHE}}$  in the FHE semantic security game.

The adversary  $\mathcal{A}_{\text{FHE}}$ :

1.  $\mathcal{A}_{\text{FHE}}$  receives a public key  $\text{pk}$  generated by the FHE challenger as  $(\text{pk}, \text{sk}) \leftarrow \text{Gen}(1^k)$ .
2. Sample random values  $a, b, v \leftarrow \{0, 1\}^n$ . Submit  $(a, 0)$  as the message pair for the FHE challenge. In response,  $\mathcal{A}_{\text{FHE}}$  receives a ciphertext  $c$  generated either as  $c \leftarrow \text{Enc}(\text{pk}, a)$  or  $\text{Enc}(\text{pk}, 0)$ .
3.  $\mathcal{A}_{\text{FHE}}$  interacts with the obfuscation extractor  $E$ , simulating black-box access to the pair of circuits  $C_0 = C_{k,a,b,v,\text{pk},\text{sk},c}$  and  $C_1 = C_{k,a,b,0,\text{pk},\text{sk},c}$ , with the challenge ciphertext  $c$ . More specifically,  $\mathcal{A}_{\text{FHE}}$  answers queries as follows (for both circuits):
  - Query at 0: output the pair  $(\text{pk}, c)$ .
  - Query at  $a$ : End the simulated interaction, and output guess ‘ $a$ ’ in the FHE challenge.
  - For any other query value: output 0.
4. If  $E$  queried either circuit on input  $a$ , then  $\mathcal{A}_{\text{FHE}}$  outputs ‘ $a$ ’ as his guess in the FHE challenge. Otherwise,  $\mathcal{A}_{\text{FHE}}$  outputs ‘0’ as his guess.

Denote by  $B$  the set of inputs  $\{\hat{b} : \text{Dec}(\text{sk}, \hat{b}) = b\}$ .

Consider first the case that  $c$  was generated as an encryption of 0. Then the view of  $E$ , consisting of only auxiliary input  $z = \emptyset$  and the “freebie” values  $(\text{pk}, c)$ , is information theoretically *independent* of both  $a$  and  $b$ . Thus, the probability that  $E$  will query either circuit on input  $a$  or  $x \in B$  is bounded by  $2^{-N} + 2^{-n}$ , which is negligible in  $n$ . In particular, the probability of  $\mathcal{A}_{\text{FHE}}$  outputting ‘ $a$ ’ in the FHE challenge in this case is negligible.

Now, consider the case that  $c$  was generated as an encryption of  $a$ . We know that if  $E$  is given correct black-box access to  $C_0$  and  $C_1$ , then he will succeed with noticeable probability  $1/q(k)$  in extracting an input on which they differ (namely, an input  $x \in B$ ). By the same argument as above, the probability of  $E$  querying either circuit on an input  $x \in B$  *before* making a query on input  $a$  is negligible, since the view of  $E$  up to this point is information theoretically independent of  $b$ . But,  $\mathcal{A}_{\text{FHE}}$ ’s simulation of black-box access to  $C_0$  and  $C_1$  is perfect up until this point. Thus, it must be that  $E$  queries one of the circuits on input  $a$  with noticeable probability  $1/q(k)$  in this case, and so  $\mathcal{A}_{\text{FHE}}$  outputs ‘ $a$ ’ in the FHE challenge with noticeable probability, breaking security of the FHE.