Indistinguishability Obfuscation from Functional Encryption for Simple Functions

Prabhanjan Ananth* Abhishek Jain† Amit Sahai ‡

Abstract

We show how to construct indistinguishability obfuscation (iO) for circuits from any non-compact functional encryption (FE) scheme with sub-exponential security against unbounded collusions. We accomplish this by giving a generic transformation from any such FE scheme into a compact FE scheme. By composing this with the transformation from sub-exponentially secure compact FE to iO (Ananth and Jain [CRYPTO'15], Bitansky and Vaikuntanathan [FOCS'15]), we obtain our main result.

Our result provides a new pathway to iO.

We use our technique to identify a *simple* function family for FE that suffices for our general result. We show that the function family $\mathcal{F}_{\text{simple}}$ is complete, where every $f_{\text{simple}} \in \mathcal{F}_{\text{simple}}$ consists of three evaluations of a Weak PRF followed by finite operations. We believe that this may be useful for realizing iO from weaker assumptions in the future.

*Center for Encrypted Functionalities and Department of Computer Science, UCLA. prabhanjan@cs.ucla.edu
This work was partially supported by grant #360584 from the Simons Foundation.

†Department of Computer Science, John Hopkins University. abhishek@cs.jhu.edu. Research supported in part from a DARPA/ARL SAFEWARE award and NSF CNS-1414023.

‡Center for Encrypted Functionalities and Department of Computer Science, UCLA. sahai@cs.ucla.edu.
Research supported in part from a DARPA/ONR PROCEED award, a DARPA/ARL SAFEWARE award, NSF Frontier Award 1413955, NSF grants 1228984, 1136174, 1118096, and 1065276, a Xerox Faculty Research Award, a Google Faculty Research Award, an equipment grant from Intel, and an Okawa Foundation Research Grant. This material is based upon work supported by the Defense Advanced Research Projects Agency through the U.S. Office of Naval Research under Contract N00014-11-1-0389. The views expressed are those of the author and do not reflect the official policy or position of the Department of Defense, the National Science Foundation, or the U.S. Government.

1 Introduction

Program obfuscation [Had00, BGI+01] concerns with the problem of making a computer program "unintelligible" while still preserving its functionality. While general-purpose program obfuscation remained an elusive goal for several decades, Garg et al [GGH+13b] changed this picture by providing the first candidate construction of indistinguishability obfuscation ($i\mathrm{O}$). Since then, $i\mathrm{O}$ has found tremendous appeal in cryptography by enabling several advanced cryptographic goals such as deniable encryption [SW14], functional encryption [GGH+13b], round-optimal secure computation [GGHR14], digital watermarking [NW15, CHV15], time-lock puzzles [BGJ+15], and more.

While the research direction of using $i\mathrm{O}$ to build other cryptographic primitives has met much success, building $i\mathrm{O}$ itself from standard cryptographic assumptions has so far proven to be notoriously difficult. The security of candidate $i\mathrm{O}$ constructions in the works of [GGH+13b, BGK+14, BR14, AGIS14, Zim15, SZ14, AB15] is only proven in generic models and lacks a reduction in the standard model. The recent works of Pass et al. [PST14] and Gentry et al. [GLSW15] provide the first standard model reductions based on different assumption on multilinear maps (see [PST14] and [GLSW15] for the description of their respective assumptions).

However, *all* of these candidates rely on the same core cryptographic primitive, namely, multilinear maps [GGH13a, CLT13, GGH15, CLT15]. This situation is unsatisfactory in light of several recent attacks on [CHL+15, GHMS14, BWZ14, CLT14, CGH+15] on most multilinear maps candidates known so far.[1]

Very recently, the independent works of Ananth and Jain [AJ15] and Bitansky and Vaikuntanathan [BV15a] presented a new direction for building $i\mathrm{O}$. They give a transformation via functional encryption [SW05, BSW11], specifically transforming *compact* public-key functional encryption (FE) for NC^1 to $i\mathrm{O}$ for P/Poly. Very roughly, in a compact FE scheme, the running time of the encryption algorithm must only have a sublinear dependence on the size complexity of functions from the function family supported by the scheme. However, presently, the only known FE constructions that achieve this compactness property are based on $i\mathrm{O}$ [GGH+13b, Wat14]. As such, this approach, so far, has not yielded any new $i\mathrm{O}$ candidates.

1.1 This Work

In this work, we continue to explore this line of research. Our vision is to progressively reduce compact FE to weaker primitives, eventually culminating in its realization from standard cryptographic assumptions. While we do not know how to fully realize this vision yet, we make some progress in this work.

We focus on weakening the requirements on FE for building $i\mathrm{O}$. In particular, we consider the following two goals: (a) While [AJ15, BV15a] require FE for NC^1 to build $i\mathrm{O}$, we set out to identify the simplest, concrete function family for FE that suffices for building $i\mathrm{O}$. (b) Further, we aim to remove the "compactness" requirement from FE for realizing $i\mathrm{O}$. We now proceed to describe our results.

$i\mathrm{O}$ from FE for "simple" functions. We consider the function family $\mathcal{F}_{\text{simple}}$ where every $f_{\text{simple}} \in \mathcal{F}_{\text{simple}}$ corresponds to three weak PRF evaluations followed by some simple, finite operations. See Figure 1 for an illustration of f_{simple} and Figure 3 in Section 4 for a formal description.

Our main result is that collusion-resistant FE for $\mathcal{F}_{\text{simple}}$ is sufficient to obtain $i\mathrm{O}$ for P/Poly.

[1]To further emphasize this point, we remark that while [GLSW15] provides a reduction to a concrete assumption on multilinear maps, no instantiation of multilinear maps where their assumption can be conjectured to hold is presently known.

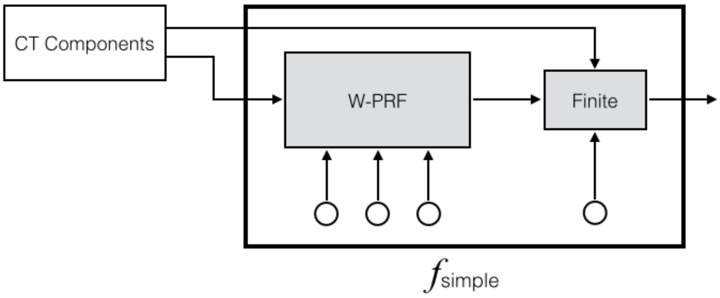

Figure 1 A figure illustrating the functionalities for which keys are needed to obtain $i\mathcal{O}$. The circles represent the constants found in the function key, that are fed to the components in boxes. The box labeled "Finite" represents a finite (constant) set of operations.

Theorem 1 (Informal) *Public-key functional encryption for $\mathcal{F}_{\text{simple}}$ with sub-exponential indistinguishability security in the selective model against unbounded collusions implies indistinguishability obfuscation for P/Poly.*

We can, in fact, further simplify $\mathcal{F}_{\text{simple}}$ such that each function in $\mathcal{F}_{\text{simple}}$ corresponds to a "local" evaluation of a polynomial-stretch PRG followed by some simple, finite operations.[2]

We stress that Theorem 1 holds even if the FE scheme is *not* compact. Therefore, in addition to simplifying the class of functions for FE required to obtain $i\mathcal{O}$, we also remove the compactness requirement from FE.

Compact FE from Non-Compact Collusion-Resistant FE. The key step underlying the above result is an unconditional transformation from any (not necessarily compact) public-key FE for NC^1 that is secure against unbounded collusions (w.r.t. selective indistinguishability security definition of [BSW11]) to a selective-secure *compact* public-key FE for NC^1.[3] The resulting scheme also inherits security against unbounded collusions. While our focus is on the public-key setting, our transformation also works in the secret-key setting. That is, if we start with a secret-key collusion-resistant FE scheme, then we obtain a secret-key compact FE scheme.

Our transformation relies on a specific form of randomized encodings [IK00, AIK04, AIK06] that we refer to as *program-decomposable* randomized encodings. By instantiating our transformation with a variant of the randomized encoding of Kilian [Kil88] (which we show satisfies the program-decomposability property) and then combining it with the compact FE to $i\mathcal{O}$ transformation of [AJ15, BV15a], we obtain Theorem 1.

Bootstrapping $i\mathcal{O}$. As an application of Theorem 1, we prove a bootstrapping theorem for $i\mathcal{O}$. Specifically, we define a different function family $\mathcal{F}$, similar at a high level to $\mathcal{F}_{\text{simple}}$. We then show that $i\mathcal{O}$ for $\mathcal{F}$ implies $i\mathcal{O}$ for P/Poly. (See Appendix A for a description of $\mathcal{F}$.)

Theorem 2 (Informal) *Indistinguishability obfuscation for $\mathcal{F}$ with sub-exponential indistinguishability security implies indistinguishability obfuscation for P/Poly.*

Previously, [GIS+10, BCG+11, App14] established bootstrapping theorems for virtual black-box obfuscation [BGI+01]. The work of [GGH+13b] gave the first bootstrapping theorem for

[2]This approach only yields compact FE with bounded collusion-resistance, but it suffices for constructing $i\mathcal{O}$. See Remark 4 in Section 3 for details.

[3]There exist general bootstrapping theorems that transform any compact FE scheme for NC^1 into a compact FE scheme that supports circuits in P/Poly [GVW12, ABSV15]. Hence for simplicity, we only focus on NC^1.

$i\mathcal{O}$, reducing $i\mathcal{O}$ for general functions to $i\mathcal{O}$ for NC^1 assuming fully-homomorphic encryption. Subsequently, Canetti et al [CLTV15] showed a similar bootstrapping theorem assuming sub-exponentially secure puncturable PRFs computable in NC^1. Recently, Bitansky et al. [BGL+15] established the first bootstrapping theorem for $i\mathcal{O}$ where obfuscating a circuit of size k allows the obfuscation of all circuits of size any polynomial in k. Our bootstrapping theorem can be viewed as an alternative proof to their result (in particular, our functionality F is different from their work).

Concurrent Work. In a concurrent and independent work, [BV15b][4] show how to build $i\mathcal{O}$ from the GGHZ assumption [GGHZ14] on composite order multilinear maps (see [GGHZ14] for a formal description of their assumption). They obtain this result via a similar approach as in our work, namely, they show a transformation from non-compact collusion-resistant FE to compact FE.

While our focus is on simplifying the requirements on FE for building $i\mathcal{O}$, we note that we can also use Theorem 1 to obtain this result by instantiating the non-compact FE scheme with the FE scheme of Garg et al [GGHZ14].

Other Related Work. Initial definitional works on FE [BSW11, O'N10] primarily considered the fully collusion-resistant setting where the adversary can obtain any polynomial number of decryption keys. However, FE for general circuits was also considered and achieved in the setting where the adversary can only obtain a single decryption key [SS10]. This was later generalized to the bounded-key setting [GVW12]. Both these schemes achieved non-compact FE. Goldwasser et al. [GKP+13] made partial progress towards realizing the goal of making such FE schemes compact. They achieve a single-key FE scheme for single-bit-output functions, where the size of the ciphertexts depend only on the depth of the functions. Note however, that while none of these schemes require $i\mathcal{O}$ or multilinear maps, we do not currently know how to achieve $i\mathcal{O}$ starting with these works, due to the limitations discussed in this paragraph.

2 Preliminaries

Throughout the paper, we denote the security parameter by λ. We assume that the reader is familiar with basic cryptographic concepts.

Given a PPT sampling algorithm A, we use $x \xleftarrow{\$} A$ to denote that x is the output of A when the randomness is sampled from the uniform distribution.

2.1 Indistinguishability Obfuscation

Here we recall the notion of indistinguishability obfuscation ($i\mathcal{O}$) that was defined by Barak et al. [BGI+01].

Definition 1 (Indistinguishability Obfuscator ($i\mathcal{O}$)) *A uniform PPT algorithm* $i\mathcal{O}$ *is called an indistinguishability obfuscator for a circuit class* $\{\mathcal{C}_\lambda\}$*, where* $\mathcal{C}_\lambda$ *consists of circuits* C *of the form* $C : \{0,1\}^\lambda \to \{0,1\}$*, if the following holds:*

- **Completeness:** *For every* $\lambda \in \mathbb{N}$*, every* $C \in \mathcal{C}_\lambda$*, every input* $x \in \{0,1\}^\lambda$ *(i.e., it belongs to the input space of* C*), we have that*

$$\mathsf{Pr}[C'(x) = C(x) \ : \ C' \leftarrow i\mathcal{O}(\lambda, C)] = 1.$$

[4] This refers to a revision of their ePrint paper.

- **Indistinguishability:** *For any PPT distinguisher D, there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that the following holds: for all sufficiently large $\lambda \in \mathbb{N}$, for all pairs of circuits $C_0, C_1 \in \mathcal{C}_\lambda$ such that $C_0(x) = C_1(x)$ for all inputs x, we have:*

$$\left|\mathsf{Pr}[D(i\mathrm{O}(\lambda, C_0)) = 1] - \mathsf{Pr}[D(i\mathrm{O}(\lambda, C_1)) = 1]\right| \leq \mathsf{negl}(\lambda)$$

2.2 Public-Key Functional Encryption

Syntax. Let $\mathcal{X} = \{\mathcal{X}_\lambda\}_{\lambda\in\mathbb{N}}$ and $\mathcal{Y} = \{\mathcal{Y}_\lambda\}_{\lambda\in\mathbb{N}}$ be ensembles where each $\mathcal{X}_\lambda$, $\mathcal{Y}_\lambda$ are sets of size, functions in λ. Let $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda\in\mathbb{N}}$ be an ensemble where each $\mathcal{F}_\lambda$ is a finite collection of functions. Each function $f \in \mathcal{F}_\lambda$ takes as input a string $x \in \mathcal{X}_\lambda$ and outputs $f(x) \in \mathcal{Y}_\lambda$.

A public-key functional encryption (FE) scheme FE for $\mathcal{F}$ consists of four algorithms ($\mathsf{FE.Setup}$, $\mathsf{FE.KeyGen}$, $\mathsf{FE.Enc}$, $\mathsf{FE.Dec}$):

- **Setup.** $\mathsf{FE.Setup}(1^\lambda)$ is a PPT algorithm that takes as input a security parameter λ and outputs a public key, (master) secret key pair ($\mathsf{FE.pk}, \mathsf{FE.msk}$).
- **Key Generation.** $\mathsf{FE.KeyGen}(\mathsf{FE.msk}, f)$ is a PPT algorithm that takes as input a master secret key $\mathsf{FE.msk}$ and a function $f \in \mathcal{F}_\lambda$ and outputs a functional key $\mathsf{FE}.sk_f$.
- **Encryption.** $\mathsf{FE.Enc}(\mathsf{FE.pk}, x)$ is a PPT algorithm that takes as input a public key $\mathsf{FE.pk}$ and a message $x \in \mathcal{X}_\lambda$ and outputs a ciphertext ct.
- **Decryption.** $\mathsf{FE.Dec}(\mathsf{FE}.sk_f, \mathsf{ct})$ is a deterministic algorithm that takes as input a functional key $\mathsf{FE}.sk_f$ and a ciphertext ct and outputs a string $y \in \mathcal{Y}_\lambda$.

Correctness. There exists a negligible function $\mathsf{negl}(\cdot)$ such that for all sufficiently large $\lambda \in \mathbb{N}$, for every message $x \in \mathcal{X}_\lambda$, and for every function $f \in \mathcal{F}_\lambda$,

$$\Pr\left[f(m) \leftarrow \mathsf{FE.Dec}(\mathsf{FE.KeyGen}(\mathsf{FE.msk}, f), \mathsf{FE.Enc}(\mathsf{FE.pk}, m))\right] \geq 1 - \mathsf{negl}(\lambda)$$

where $(\mathsf{FE.pk}, \mathsf{FE.msk}) \leftarrow \mathsf{FE.Setup}(1^\lambda)$, and the probability is taken over the random coins of all algorithms.

Selective Security. We recall indistinguishability-based selective security for FE. This security notion is modeled as a game between the challenger and the adversary where the adversary can request functional keys and ciphertexts from the challenger. Specifically, the adversary can submit function queries f to the challenger and receive corresponding functional keys. It can also submit a message query of the form (x_0, x_1) and in response, the challenger encrypts message x_b and sends the ciphertext back to the adversary. The adversary wins the game if she can guess b with probability significantly greater than $1/2$ and if $f(x_0) = f(x_1)$ for all function queries f. The only constraint here is that the adversary has to declare the challenge messages at the beginning of the game itself.

Definition 2 (IND-secure FE) *A public-key functional encryption scheme* $\mathsf{FE} = (\mathsf{FE.Setup}, \mathsf{FE.KeyGen}, \mathsf{FE.Enc}, \mathsf{FE.Dec})$ *for a function family $\mathcal{F}$ is said to be q_{key}-selectively secure if for any PPT adversary $\mathcal{A}$, for all sufficiently large $\lambda \in \mathbb{N}$, the advantage of $\mathcal{A}$ is*

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{FE}} = \left|\mathsf{Pr}[\mathsf{Expt}_{\mathcal{A}}^{\mathsf{FE}}(1^\lambda, 0) = 1] - \mathsf{Pr}[\mathsf{Expt}_{\mathcal{A}}^{\mathsf{FE}}(1^\lambda, 1) = 1]\right| \leq \mathsf{negl}(\lambda),$$

where for each $b \in \{0, 1\}$ and $\lambda \in \mathbb{N}$, the experiment $\mathsf{Expt}_{\mathcal{A}}^{\mathsf{FE}}(1^\lambda, b)$ is defined as follows:

1. ***Challenge message query***: *$\mathcal{A}$ submits a message pair (x_0, x_1) to C.*

2. *The challenger C computes* $(\mathsf{FE.pk}, \mathsf{FE.msk}) \leftarrow \mathsf{FE.Setup}(1^\lambda)$ *and sends* $\mathsf{FE.pk}$ *to the adversary. It then computes* $\mathsf{ct} = \mathsf{FE.Enc}(\mathsf{FE.msk}, x_b)$ *and sends* ct *to $\mathcal{A}$.*

3. ***Function queries***: *The following is repeated up to q_{key} times: $\mathcal{A}$ submits a function query $f \in \mathcal{F}_\lambda$ to C. The challenger C computes the function key* $\mathsf{FE}.sk_f \leftarrow \mathsf{FE.KeyGen}(\mathsf{FE.msk}, f)$ *and sends it to $\mathcal{A}$.*

4. *If there exists a function query f such that $f(x_0) \neq f(x_1)$, then the output of the experiment is $\perp$. Otherwise, the output of the experiment is b', where b' is the output of $\mathcal{A}$.*

Remark 1 (Selective-security against unbounded collusions) *One can consider a strengthening of the above definition where the adversary is allowed to make any unbounded polynomial number of function queries. We refer to this as selective security against unbounded collusions.*

2.2.1 Compactness

We now recall the notion of *compact* FE from [AJ15]. In a compact FE scheme, the running time of the encryption algorithm only depends on the security parameter and the input message length. In particular, it is independent of the complexity of the function family supported by the FE scheme.

Definition 3 (Compact FE) *Let $p(\cdot)$ be a polynomial. A selectively secure public-key FE scheme* $\mathsf{FE} = (\mathsf{FE.Setup}, \mathsf{FE.KeyGen}, \mathsf{FE.Enc}, \mathsf{FE.Dec})$, *defined for an input space $\mathcal{X} = \{\mathcal{X}_\lambda\}$ and function space $\mathcal{F} = \{\mathcal{F}_\lambda\}$ is said to be* ***compact*** *if for all $\lambda \in \mathbb{N}$, the running time of the encryption algorithm* $\mathsf{FE.Enc}$, *on input* 1^λ, $\mathsf{FE.pk}$ *and a message $x \in \mathcal{X}_\lambda$, is $p(\lambda, q_{\mathsf{key}}, |x|)$.*

Remark 2 (Sublinear dependence) *As observed in [BV15a], a milder form of compact FE where the running time of the encryption algorithm is sublinear in the size of any $f \in \mathcal{F}$ is sufficient to obtain iO. For simplicity of exposition, however, we will use Definition 3 in this manuscript. Indeed, our main transformation presented in Section 3 holds for this stronger definition.*

3 Compact FE from Collusion-Resistant FE

We give a generic transformation from any collusion-resistant FE scheme (CRFE) that is *not* necessarily compact to a compact FE (CFE) scheme. We start by giving an overview of our approach in Section 3.1. Next, in Section 3.2 we define the notion of program decomposable randomized encodings that is used in our transformation. We then describe our transformation in Section 3.3 followed by its security proof in Section 3.4

3.1 Overview

Recall that in a non-compact FE scheme, the running time of the encryption algorithm may depend polynomially on the complexity of functions from the function family supported by the scheme. Our goal is to remove this dependence generically, i.e., give a transformation from any non-compact FE scheme into a compact FE scheme where the running time of the encryption algorithm is *independent* of the function family.

Towards that end, let us first recall some related results known in the literature and folklore. The works of [GVW12, ABSV15] show that dependence on the depth of the function can be removed generically by using any (efficient) randomized encoding [IK00, AIK04, AIK06]. Recall that a randomized encoding of any polynomial-sized circuit can be computed in only logarithmic depth. Given this observation, their transformation is simple: instead of issuing a key for a function f, we issue a key for a function f' where f' takes input x for f and computes an RE of (f, x). Indeed, this idea has been used extensively throughout cryptography (see [App11] for a survey).

Next, we note the folklore observation that dependence on the function output-size can also be removed generically if the underlying FE scheme is *collusion-resistant*. Very roughly, if a function f has ℓ-bit outputs, then the idea is to issue ℓ different keys $K_{f_1}, \ldots, K_{f_\ell}$, where K_{f_i} corresponds to computing the ith bit of the output of f. Now, since each function key only corresponds to binary functions, the resulting scheme will not have output-size dependence.

Of course, in a non-compact FE scheme, the running time of the encryption algorithm may grow polynomially with the *size* of the function (in particular, the number of gates in the circuit representation of the function). As such, the above simple transformations do not suffice for our purpose.

Our Idea. Towards that end, our main idea is to further leverage collusion-resistance to achieve compactness generically. Very roughly, instead of issuing a key for a circuit C, our idea is to issue multiple keys $K_{G_1}, \ldots, K_{G_\ell}$ where each key K_{G_i} corresponds to computing a "small component" of C. From an efficiency viewpoint, we require that ℓ is some fixed polynomial in the security parameter and for every i, the size of the circuit G_i computed by K_{G_i} is *independent* of $|C|$. From a security viewpoint, intuitively, we want that given a key set $K_{G_1}, \ldots, K_{G_\ell}$ corresponding to a circuit C and an encryption of a message x, an evaluator should only learn $C(x)$.

Put differently, on the one hand, we want to "decompose" the process of computing $C(x)$ into ℓ different parts such that the size of each part is independent of $|C|$. At the same time, these parts should be "tied" together in such a manner that when put together, they reveal $C(x)$, and nothing else otherwise. A natural approach to achieve the efficiency requirement is to simply let G_i correspond to evaluating the ith gate of C. Note, however, that this must be done in a manner that preserves the security of the FE.

Program-Decomposable Randomized Encodings. To address this problem, we once again turn to randomized encodings. Our idea is to use a specific form of RE that we refer to as *program decomposable* RE (PD-RE). In a PD-RE scheme, the encoding process includes a "decomposition" process for programs[5] that decomposes a circuit C into several parts $C_1, \ldots, C_\ell$, where for every i, we have that $|C_i|$ is independent of $|C|$. For every i, the encoding algorithm *Enc* on input C_i and x outputs an encoding $\widehat{C_i, x}$. The decoding algorithm takes as input the tuple $\{\widehat{C_i, x}\}$ and should output $y = C(x)$. The efficiency requirement is that for every i, $|Enc|$ is independent of $|C|$. The security requirement, however, is still the same as in standard RE: for any $C = C_1, \ldots, C_\ell$ and input pair (x^0, x^1), the encodings $(\widehat{C_1, x^0}, \ldots, \widehat{C_\ell, x^0})$ and $(\widehat{C_1, x^1}, \ldots, \widehat{C_\ell, x^1})$ are computationally indistinguishable as long as $C(x^0) = C(x^1)$.

We observe that many RE schemes from the literature directly yield PD-RE schemes. For example, Yao's garbling technique [Yao86] yields a PD-RE scheme for general circuits. For any circuit C, C_i corresponds to its ith gate and *Enc* on input (C_i, x) corresponds to computing the ith "garbled gate table." Later, we also identify another PD-RE scheme by modifying the RE scheme of Kilian [Kil88] (which in turn uses Barrington's theorem [Bar86]). This allows

[5]For concreteness, we will restrict our discussion to circuits here, although this notion is compatible with other computing models such as branching programs. See Section 3 for details.

us to identify a simple, concrete function family for collusion-resistant FE that suffices for our transformation. (See Section 4 for details.)

Given such a PD-RE scheme, we obtain a compact FE construction as follows: a key for a circuit C consists of a key set $\{K_{G_i}\}_{i\in\ell}$. For every i, the function G_i associated with the key K_i takes as input a message x and computes $Enc(C_i, x)$. Note that this is a randomized procedure, and that the randomness among different evaluations must be correlated. (We address this further below.) An evaluator who is given a key set $\{K_{G_i}\}_{i\in\ell}$ and an encryption ct of x can now first perform ℓ FE decryptions of ct (one with each key K_i) to obtain a PD-RE $(\widehat{C_1, x}, \ldots, \widehat{C_\ell, x})$ of (C, x). Next, it can perform the RE decoding procedure to obtain $C(x)$. From the security of PD-RE, we are guaranteed that the evaluator cannot learn anything else.

We remark that the "program decomposability" property of garbled circuits has been used in many cryptographic schemes in the past. To the best of our knowledge, the first such use is due to Beaver et al. [BMR90] who used garbled circuits to construct constant-round multiparty computation protocols. We note, however, that in their construction, they only use the fact that each gate table can be computed in constant *depth*; for our purposes, it is important that the *size* of each decomposed unit is independent of the total size of the circuit. This property of garbled circuits was recently used by Bitansky et al. [BGL+15] in their construction of succinct REs.

More Details. We now provide some more details of the above construction. First note that the program encoding procedure is a randomized functionality. To provide randomness to each invocation of the program encoding procedure Enc, as an initial "straw man" proposal, we consider the following: we can modify the encryption algorithm of FE scheme to additionally encrypt a random key K for a weak pseudo-random function (PRF) along with the input message x. The function G_i computed by the key K_{G_i} now consists of the following steps: on input (K, x), it first evaluates K on a random tag hardwired in its description to obtain r_i. Next, it computes and outputs $Enc(C_i, x)$ using randomness r_i.

We highlight a couple of problems with the above approach: first, we note that different invocations of program encoding procedure Enc of known PD-RE schemes critically use *correlated* randomness – intuitively, this use of correlated randomness is needed to make different parts of program decomposition "talk with each other" when decoding. We address this need for correlation by explicitly considering set systems that capture the necessary correlations, and incorporating this into our construction: we refer the reader to Section 3 for details. A more important problem is that we cannot directly rely upon the standard security of the underlying FE scheme to prove the security of the new scheme. This is because Enc is a randomized functionality whereas standard FE only considers *deterministic* functions. The recent work of [GJKS15] studies the problem of public-key FE for randomized functions; however, they give a specific construction using iO which is therefore not suitable for our purposes.[6]

To address this problem, we apply the "trapdoor circuits" technique of De Caro et al [CIJ+13]. Very roughly, we modify G_i such that it works in two modes: in the "honest" mode, it performs the same functionality as discussed above. In the "trapdoor" mode, it outputs a fixed value hardwired in its description. Using this idea, in our proof, we can switch from honest computation of Enc_i to the PD-RE simulator. This step only relies on the security of the underlying PD-RE. Now, we can simply change the message x_0 in the ciphertext to x_1 by relying upon the security of the underlying FE scheme. Finally, we change G_i again to the honest mode, completing the proof. We remark that several recent works [GGHR14, BS15, ABSV15]) make a similar use of the trapdoor circuits technique in the context of FE.

[6] In the secret-key setting, [KSY15] show a generic transformation from any secret-key FE for deterministic functions into one that supports randomized functions. However, no such transformation is known in the public-key setting.

3.2 Program Decomposable Randomized Encodings

Let x be a string of length ℓ and let S be a subset of $[\ell]$. We define $x_{|S}$ to be the string that is obtained by concatenating all the bits of x corresponding to positions in S. We refer to this as "x being restricted to S".

Syntax. A Program Decomposable RE, defined for a function family $\mathcal{F}$, consists of a tuple of algorithms $(\mathsf{Decomp}, \mathsf{PrgEnc}, \mathsf{Dec})$ described below:

- *Program Decomposition*: Let f be a function in $\mathcal{F}$ with input length ℓ_{inp}. The deterministic algorithm Decomp takes as input security parameter λ, a description of f and performs the following steps:
  1. Compute a set of program components $\mathsf{P} = (P_1, \ldots, P_{\ell_{\text{prg}}})$ representing f.
  2. Generate two set systems $\mathcal{S} = \{S_1, \ldots, S_{\ell_{\text{prg}}}\}$ and $\mathcal{I} = \{I_1, \ldots, I_{\ell_{\text{prg}}}\}$, where $S_i \subseteq [\ell_R], I_i \subseteq [\ell_{\text{inp}}]$ for all $i \in [\ell_{\text{prg}}]$, and ℓ_R is a polynomial in $(\lambda, \ell_{\text{prg}})$.
  3. Output $(\mathsf{P}, \mathcal{S}, \mathcal{I}, \ell_R)$.
- *Program Encoding*: Let $\mathsf{P} = P_1, \ldots, P_{\ell_{\text{prg}}}$ be a program decomposition and $x = x_1, \ldots, x_{\ell_{\text{inp}}}$ be an input for P that we wish to encode. Let $\mathcal{S} = \{S_1, \ldots, S_{\ell_{\text{prg}}}\}$ and $\mathcal{I} = \{I_1, \ldots, I_{\ell_{\text{prg}}}\}$ be two set systems with $S_i \subseteq [\ell_R], I_i \subseteq [\ell_{\text{inp}}]$. Let r be a string of length ℓ_R chosen uniformly at random.

  PrgEnc is a PPT algorithm that takes as input a program component P_i, string $x_{|I_i}$, random string $r_{|S_i}$ and outputs an encoding $\widehat{P_i, x}$.
- *Output Decoding*: Dec is a deterministic polynomial-time algorithm that takes as input an encoding tuple $\left(\left\{\widehat{P_i, x}\right\}_{i \in [\ell_{\text{prg}}]}\right)$ and outputs a value y.

This completes the description of the algorithms of a program-decomposable RE. We now state our efficiency requirements and then formally define correctness and security of PD-RE.

Efficiency. On any input f, we require the output $\Big(\mathsf{P} = \{P_i\}_{i \in \ell_{\text{prg}}}, \mathcal{S} = \{S_i\}_{i \in \ell_{\text{prg}}}, \mathcal{I} = \{I_i\}_{i \in \ell_{\text{prg}}}\Big)$ of the Decomp algorithm to be such that:

- For every $i \in [\ell_{\text{prg}}]$, $|P_i| = p(\lambda)$ where $p(\cdot)$ is a fixed polynomial that is independent of $|P|$.
- Every set in $\mathcal{S}$ and $\mathcal{I}$ is of size $q = q(\lambda)$, where $q(\cdot)$ is a fixed polynomial that is independent of $|P|$.

A direct consequence of the above two properties is that the running time of PrgEnc is $t(\lambda)$, where $t(\cdot)$ is a fixed polynomial that is independent of $|P|$.

Correctness. We say that a program decomposable RE $(\mathsf{Decomp}, \mathsf{PrgEnc}, \mathsf{Dec})$ for $\mathcal{F}$ is *correct* if for every $f \in \mathcal{F}$ and input x to f, we have the following quantity to be at least $\mathsf{negl}(\lambda)$:

$$\Pr\left[\mathsf{Dec}\left(\mathsf{PrgEnc}\left(P_1, x_{|I_1}; r_{|S_1}\right), \ldots, \mathsf{PrgEnc}\left(P_{\ell_{\text{prg}}}, x_{|I_{\ell_{\text{prg}}}}; r_{|S_{\ell_{\text{prg}}}}\right)\right) = P(x)\right]$$

where $\left(\mathsf{P}, \mathcal{S} = \left\{S_1, \ldots, S_{\ell_{\text{prg}}}\right\}, \mathcal{I} = \left\{I_1, \ldots, I_{\ell_{\text{prg}}}\right\}, \ell_R\right) \leftarrow \mathsf{Decomp}(f)$, and r is a string of length ℓ_R picked uniformly at random.

Security. We say that a program decomposable RE $(\mathsf{Decomp}, \mathsf{PrgEnc}, \mathsf{Dec})$ for $\mathcal{F}$ is *secure* if for every PPT adversary $\mathcal{A}$, every $f \in \mathcal{F}$ and input pair (x^0, x^1) such that $f(x^0) = f(x^1)$, we have that:

$$\left| \Pr\left[\mathcal{A}\left(\widehat{P_1, x^0}, \ldots, \widehat{P_{\ell_{\text{prg}}}, x^0} \right) = 1 \right] - \Pr\left[\mathcal{A}\left(\widehat{P_1, x^1}, \ldots, \widehat{P_{\ell_{\text{prg}}}, x^1} \right) = 1 \right] \right| = \mathsf{negl}(\lambda),$$

where $\widehat{P_i, x^b} \leftarrow \mathsf{PrgEnc}\left(P_i, x^b_{|I_i}; r_{|S_i} \right)$, $\left(\mathsf{P}, \mathcal{S} = \{S_1, \ldots, S_{\ell_{\text{prg}}}\}, \mathcal{I} = \{I_1, \ldots, I_{\ell_{\text{prg}}}\}, \ell_R \right) \leftarrow \mathsf{Decomp}(f)$, and r is a random string of length ℓ_R.

3.3 Our Transformation: From CRFE to CFE

Let $\mathsf{CRFE} = (\mathsf{CRFE.Setup}, \mathsf{CRFE.KeyGen}, \mathsf{CRFE.Enc}, \mathsf{CRFE.Dec})$ be any public-key FE scheme for a function family $\mathcal{F}_{\mathsf{CRFE}}$ that is selective-secure against unbounded collusions. We defer the description of $\mathcal{F}_{\mathsf{CRFE}}$ to later. Given CRFE, we construct a *compact* public-key FE scheme $\mathsf{CFE} = (\mathsf{CFE.Setup}, \mathsf{CFE.KeyGen}, \mathsf{CFE.Enc}, \mathsf{CFE.Dec})$ for a function family $\mathcal{F}$. The family $\mathcal{F}_\lambda = \{\mathcal{F}_\lambda\}_{\lambda \in \mathbb{N}}$ comprises of functions with input length λ. The associated message space is denoted by $\mathcal{X} = \{\mathcal{X}_\lambda\}_\lambda$, where $\mathcal{X}_\lambda = \{0,1\}^\lambda$. The resulting scheme CFE inherits the security properties of CRFE, namely, it achieves selective-security against unbounded collusions.

Our transformation uses the following additional tools:

- A program-decomposable RE scheme $\mathsf{PDRE} = (\mathsf{PDRE.Decomp}, \mathsf{PDRE.PrgEnc}, \mathsf{PDRE.Dec})$ for the function family $\mathcal{F}$.

- Weak pseudorandom function[7] family $\mathcal{PRF} = \{PRF_K(\cdot) : \{0,1\}^\lambda \to \{0,1\}\}$ and a symmetric encryption scheme $\mathsf{Sym} = (\mathsf{Sym.Setup}, \mathsf{Sym.Enc}, \mathsf{Sym.Dec})$. We assume that the ciphertexts produced by Sym are pseudorandom.

We now describe the compact FE scheme scheme CFE below.

Setup $\mathsf{CFE.Setup}(1^\lambda)$: On input security parameter λ, execute $\mathsf{CRFE.Setup}(1^\lambda)$ to obtain $(\mathsf{CRFE.MSK}, \mathsf{CRFE.PK})$. Output the master secret key $\mathsf{CFE.MSK} = \mathsf{CRFE.MSK}$ and the public key $\mathsf{CFE.PK} = \mathsf{CRFE.PK}$.

Key Generation $\mathsf{CFE.KeyGen}(\mathsf{CFE.MSK}, f)$: On input a master secret key $\mathsf{CFE.MSK} = \mathsf{CRFE.MSK}$ and a function $f \in \mathcal{F}_\lambda$,

- Execute $\mathsf{PDRE.Decomp}(f)$ to obtain $(\mathsf{P}, \mathcal{S}, \mathcal{I}, \ell_R)$. Parse $\mathsf{P} = (P_1, \ldots, P_{\ell_{\text{prg}}})$.
- Pick tags $\mathsf{tag}_1, \ldots, \mathsf{tag}_{\ell_R}$, where $\mathsf{tag}_i \in \{0,1\}^\lambda$ with $\ell_R = \text{poly}(\ell_{\text{prg}}, \lambda)$.
- Parse $\mathcal{S}$ as $\{S_1, \ldots, S_{\ell_{\text{prg}}}\}$. Assign $\mathsf{TAG}_i = (\mathsf{tag}_k)_{k \in S_i}$ for $i \in [\ell_{\text{prg}}]$.
- Pick strings $CT_1, \ldots, CT_{\ell_{\text{prg}}}$ uniformly at random. (The length of CT_i will be clear later).
- Execute $\mathsf{CRFE.KeyGen}(\mathsf{CRFE.MSK}, \mathsf{Encode}[P_i, \mathsf{TAG}_i, I_i, CT_i])$ with $i \in [\ell_{\text{prg}}]$ to obtain $\mathsf{CRFE.sk}_i$. The function $\mathsf{Encode}[\cdot, \cdot, \cdot, \cdot]$ is described in Figure 2.

Output $\mathsf{CFE.sk}_f = \left(\mathsf{CRFE.sk}_1, \ldots, \mathsf{CRFE.sk}_{\ell_{\text{prg}}} \right)$.

Encryption $\mathsf{CFE.Enc}(\mathsf{CFE.PK}, x)$: On input a public key $\mathsf{CFE.PK} = \mathsf{CRFE.PK}$ and a message x, sample a PRF key K. Execute $\mathsf{CRFE.Enc}(\mathsf{CRFE.PK}, (x, K, \perp, \mathsf{mode} = 0))$ to obtain $\mathsf{CRFE.CT}$.

[7] A weak pseudorandom function is a type of pseudorandom function wherein the adversary, in the security game, is handed evaluations of the weak PRF on random points. This is in contrast to the scenario of PRFs, where the adversary can choose his queries.

$\underline{\mathsf{Encode}[P, \mathsf{TAG}, I, CT](x, K, \mathsf{Sym}.K, \mathsf{mode})}$

If $\mathsf{mode} = 0$:

- Parse TAG as $\left(\mathsf{tag}_1, \ldots, \mathsf{tag}_p\right)$.
- Execute the weak pseudo-random function $PRF_K(\mathsf{tag}_i)$ to get strings r_i. Assign $r = r_1 || \cdots || r_p$.
- Execute $\mathsf{PDRE.PrgEnc}\left(P, x_{|I}; r\right)$ to obtain the encoding $\widehat{P, x}$. Output $\widehat{P, x}$.

If $\mathsf{mode} = 1$:

- Execute $\mathsf{Sym.Dec}(\mathsf{Sym}.K, CT)$ to obtain $\widehat{P, x}$. Output $\widehat{P, x}$.

Figure 2

Output the ciphertext $\mathsf{CFE.CT}_x = \mathsf{CRFE.CT}$.

Decryption $\mathsf{CFE.Dec}(\mathsf{CFE.sk}_f, \mathsf{CFE.CT}_x)$: On input a ciphertext $\mathsf{CFE.CT}_x = \mathsf{CRFE.CT}$ and a functional key $\mathsf{CFE.sk}_f = (\mathsf{CRFE.sk}_1, \ldots, \mathsf{CRFE.sk}_{\ell_{\mathrm{prg}}})$, execute the decryption algorithm $\mathsf{CRFE.Dec}(\mathsf{CRFE.sk}_i, \mathsf{CRFE.CT})$ to obtain $\widehat{P_i, x}$. Execute $\mathsf{PDRE.Dec}\left(\left\{\widehat{P_i, x}\right\}_{i \in [\ell_{\mathrm{prg}}]}\right)$ to obtain y. Output y.

This completes the description of the compact FE scheme.

Correctness. Observe that the output of $\mathsf{CRFE.Dec}(\mathsf{CRFE.sk}_i, \mathsf{CRFE.CT})$ is an encoding $\widehat{P_i, x}$. These encodings are "valid", meaning that they can be obtained by first running the decomposition algorithm on f and then encoding the resulting decomposed program components along with x. Therefore, by the correctness of PDRE, we have that the output of $\mathsf{PDRE.Dec}\left(\left\{\widehat{P_i, x}\right\}_{i \in \ell_{\mathrm{prg}}}\right)$ is $f(x)$.

Compactness. First observe that the run-time of $\mathsf{CFE.Enc}$ depends only on $|x|$, run-time of $\mathsf{CRFE.Enc}$ and λ. So it suffices for us to focus on $\mathsf{CRFE.Enc}$. Since we make no assumptions on the compactness of CRFE, it could very well be the case that the run-time of $\mathsf{CRFE.Enc}$ depends polynomially on the size complexity of functions in $\mathcal{F}_{\mathsf{CRFE}}$. Note, however, that the size of any $\mathsf{Encode} \in \mathcal{F}_{\mathsf{CRFE}}$ is simply a fixed polynomial in λ since it only involves weak PRF evaluations and computing PrgEnc, whose complexity is polynomial in λ. Summing up, we have that the run-time of $\mathsf{CFE.Enc}$ is $\mathsf{poly}(\lambda, |x|)$, as required.

Security. We prove the following theorem in Section 3.4:

Theorem 3 (Security of CFE) *If* CRFE *is a public-key FE with selective-security against unbounded collusions,* PDRE *is a secure PD-RE scheme and* $\mathcal{PRF}$ *is a weak PRF, we have that* CFE *is a public-key compact FE with selective-security against unbounded collusions.*

Remark 3 (Secret-key setting) *The above transformation is presented in the public-key setting. It is easy to see that the same transformation also works in the secret-key setting. Namely, if we start with a secret-key collusion-resistant FE scheme, we obtain a secret-key compact FE scheme.*

Remark 4 (Replacing weak PRF with PRG) *We can replace the weak PRF in the above approach with a particular type of PRGs. We require the property that each block in the output* *of PRG can be computed in time independent of the stretch of PRG. Now, the i^{th} program component can be encoded using (as randomness) the corresponding i^{th} block in the output of PRG. Also note that the generic symmetric-key encryption (used in mode 1) can also be instantiated with a one-time pad implemented using a standard PRG. This is because we only need one-time security of the hardwired ciphertext in our proof. Finally, we note that with this approach, the resulting scheme will not be collusion-resistant, although this does not affect the implication to* $i\mathcal{O}$.

Remark 5 (Bootstrapping theorem) *If we choose weak PRFs in* NC^1 *and suitably instantiate Program Decomposable RE (for ex., garbled circuits), Theorem 3 yields a bootstrapping mechanism for transforming a non-compact collusion-resistant FE for* NC^1 *into a compact collusion-resistant FE for P/poly (assuming DDH or LWE). This is a generalization of the bootstrapping theorem of [ABSV15] which was (non-)compactness preserving.*

3.4 Proof of Theorem 3

The main idea is to hardwire the output program encodings in the functional keys. After this, we can use the security of Program Decomposable RE to switch from one input to another. However, functional keys do not hide its associated function and hence to enable the hardwiring process, we use the trapdoor branch. We compute a symmetric encryption of the output encoding. We then switch the mode in the message to now decrypt this (symmetric) ciphertext instead of executing the encoding procedure. Once this is done, we have the program encoding hardwired as desired. We now explain the technical details.

We define the advantage of a PPT adversary $\mathcal{A}$ in $\mathsf{Hybrid}_{i.b}$ to be $\mathsf{Adv}_{\mathcal{A},i.b}$.

$\forall\, b \in \{0,1\},\ \mathsf{Hybrid}_{1.b}$: This corresponds to the real experiment when the challenger uses the challenge bit b. That is, when the adversary submits a message pair (x_0, x_1), the challenger encrypts the message x_b as part of the challenge ciphertext. The hybrid outputs whatever the adversary outputs.

$\forall\, b \in \{0,1\},\ \mathsf{Hybrid}_{2.b}$: The output of the functional keys w.r.t the challenge ciphertext is hardwired in their respective CT components.

At the beginning of the game, the challenger samples a symmetric key $\mathsf{Sym}.K$ by executing $\mathsf{Sym.Setup}$. It answers challenge message query from the adversary as in the previous hybrid. Denote the challenge ciphertext answered by the challenger to be $\mathsf{CFE.CT}^* = \mathsf{CFE.Enc}(\mathsf{CFE.PK}, (x_b, K, 0))$.

Upon receiving a function query f, the challenger first executes $\mathsf{PDRE.Decomp}(f)$ to obtain $(\mathsf{P}, \mathcal{S}, \mathcal{I}, \ell_R)$. It then picks the tags $\mathsf{tag}_1, \ldots, \mathsf{tag}_{\ell_R}$ uniformly at random with $\mathsf{tag}_i \in \{0,1\}^\lambda$. Let the set family $\mathcal{S}$ (resp., $\mathcal{I}$) be denoted by $\{S_1, \ldots, S_{\ell_{\mathrm{prg}}}\}$ (resp., $\{I_1, \ldots, I_{\ell_{\mathrm{prg}}}\}$). For $i = 1, \ldots, \ell_{\mathrm{prg}}$, it does the following:

1. Assigns $\mathsf{TAG}_i = \cup_{k \in S_i} (\mathsf{tag}_k)$ for $i \in [\ell_{\mathrm{prg}}]$. Alternately, denote $\mathsf{TAG}_i = (\tilde{\mathsf{tag}}_1, \ldots, \tilde{\mathsf{tag}}_p)$, where $p = |S_i|$.
2. Execute the weak pseudo-random function $PRF_{K_j}(\mathsf{tag}_j)$ to get strings r_j. Assign $r = r_1 || \cdots || r_p$.
3. Execute $\mathsf{PrgEnc}(P_i, \mathbf{x}_{|I_i}; r)$ to obtain the encoding $\widehat{P_i, \mathbf{x}}$, where $\mathbf{x} = x_b$. Output $\widehat{P_i, \mathbf{x}}$.
4. The challenger encrypts $\widehat{P_i, \mathbf{x}}$ by executing $\mathsf{Sym.Enc}(\mathsf{Sym}.K, \widehat{P_i, \mathbf{x}})$ to obtain $\mathsf{Sym.ct}_i$. It sets $CT_i = \mathsf{Sym.ct}_i$.
5. In the final step, it executes $\mathsf{CRFE.KeyGen}(\mathsf{CRFE.MSK}, \mathsf{Encode}[P_i, \mathsf{TAG}_i, I_i, CT_i])$ to obtain $\mathsf{CRFE.sk}_i$.

The challenger then sends across the functional key $\mathsf{CFE.sk}_f = (\mathsf{CRFE.sk}_i)_{i\in[\ell_{\mathrm{prg}}]}$ to the adversary. The challenger repeats the above process for every function query.

Lemma 1 *Assuming the security of* Sym*, we have* $|\mathsf{Adv}_{\mathcal{A},1.b} - \mathsf{Adv}_{\mathcal{A},2.b}| \leq \mathsf{negl}(\lambda)$ *for* $b \in \{0,1\}$*, where* negl *is a negligible function.*

$\forall\ b \in \{0,1\},\ \mathsf{Hybrid}_{3.b}$: The challenger changes the mode in the challenge ciphertext from $\mathsf{mode} = 0$ to $\mathsf{mode} = 1$. That is, upon receiving the message query (x_0, x_1), the challenger first samples the symmetric secret key $\mathsf{Sym}.K$ by executing Sym.Setup. It then computes the challenge ciphertext $\mathsf{CFE.CT}^* \leftarrow \mathsf{CFE.Enc}(\mathsf{CFE.PK}, (\perp, \perp, \mathsf{Sym}.K, \mathsf{mode} = 1))$. It sends $\mathsf{CFE.CT}^*$ to the adversary. The functional queries are answered as in the previous hybrid.

Lemma 2 *Assuming the selective security of* CRFE*, we have* $|\mathsf{Adv}_{\mathcal{A},2.b} - \mathsf{Adv}_{\mathcal{A},3.b}| \leq \mathsf{negl}(\lambda)$ *for* $b \in \{0,1\}$*, where* negl *is a negligible function.*

$\forall\ b \in \{0,1\},\ \mathsf{Hybrid}_{4.b}$: This hybrid is identical to $\mathsf{Hybrid}_{2.b}$ except that the randomness supplied to PDRE.PrgEnc is picked uniformly at random. Recall that the randomness in $\mathsf{Hybrid}_{2.b}$ was generated by executing weak pseudorandom functions. More precisely, we pick a random string $\mathbf{r} \in \{0,1\}^{\ell_R}$ at random. replace Bullet 2 in $\mathsf{Hybrid}_{2.b}$ with the following.

2. Set $r = \mathbf{r}_{|S_i}$.

As before, the output of this hybrid is the adversary's output.

Lemma 3 *Assuming the security of* $\mathcal{PRF}$*, we have* $|\mathsf{Adv}_{\mathcal{A},3.b} - \mathsf{Adv}_{\mathcal{A},4.b}| \leq \mathsf{negl}(\lambda)$ *for* $b \in \{0,1\}$*, where* negl *is a negligible function.*

Lemma 4 *Assuming the security of* PDRE*, we have* $|\mathsf{Adv}_{\mathcal{A},4.0} - \mathsf{Adv}_{\mathcal{A},4.1}| \leq \mathsf{negl}(\lambda)$*, where* negl *is a negligible function.*

3.5 Proof of Theorem 3 cont'd: Proofs of Lemma 1,2,3,4

Proof of Lemma 1. We construct a reduction $\mathcal{B}$ that uses $\mathcal{A}$ to break the security of Sym.

The message query by the adversary is answered by $\mathcal{B}$ as in Hybrid_1. The When $\mathcal{B}$ receives a function query of f, it first decomposes f into program components and then computes its encodings using the randomness derived from a weak PRF. This is performed as described in $\mathsf{Hybrid}_{2.b}$. We denote the resulting encodings to be $\widehat{P_i, \mathbf{x}}$, for $i \in [\ell_{\mathrm{prg}}]$, where $\mathbf{x} = x_b$. At this point, $\mathcal{B}$ submits $\{\widehat{P_i, \mathbf{x}}\}_{i\in[\ell_{\mathrm{prg}}]}$ to the challenger of Sym. In return it receives the ciphertexts $\{CT_i = \mathsf{Sym.ct}_i\}_{i\in[\ell_{\mathrm{prg}}]}$. Finally, $\mathcal{B}$ computes $\mathsf{CRFE.sk}_i \leftarrow \mathsf{CRFE.KeyGen}(\mathsf{CRFE.MSK}, \mathsf{Encode}[P_i, \mathsf{TAG}_i, I_i, CT_i])$ for $i \in [\ell_{\mathrm{prg}}]$. It then sends $\mathsf{CFE.sk}_f = \{\mathsf{CRFE.sk}_i\}_{i\in[\ell_{\mathrm{prg}}]}$ to the adversary.

If Sym's challenger answers with a random string then we are in $\mathsf{Hybrid}_{1.b}$ otherwise we are in $\mathsf{Hybrid}_{2.b}$. Thus, the advantage of $\mathcal{B}$ breaking the security of Sym is $|\mathsf{Adv}_{\mathcal{A},1.b} - \mathsf{Adv}_{\mathcal{A},2.b}|$ which is negligible in λ.

Proof of Lemma 2. We construct a reduction $\mathcal{B}$ that uses $\mathcal{A}$ to break the security of CRFE.

When the reduction receives message query (x_0, x_1), it computes the following pairs of messages:

$$\Big(\widetilde{y_0} = (x_b, K, \perp, 0),\ \widetilde{y_1} = (\perp, \perp, \mathsf{Sym}.K, 1)\Big),$$

where K is a weak PRF key and $\mathsf{Sym}.K$ is the secret key sampled using Sym.Setup. It then forwards the message query $(\widetilde{y_0}, \widetilde{y_1})$ to the challenger of CRFE. The reduction $\mathcal{B}$ then forwards the challenge ciphertext, received from the challenger of CRFE, to $\mathcal{A}$. When $\mathcal{B}$ receives a functional query f, it computes the functions $G_i = \mathsf{Encode}[P_i, \mathsf{TAG}_i, I_i, CT_i]$, for $i \in [\ell_{\mathrm{prg}}]$ as in $\mathsf{Hybrid}_{2.b}$ (or $\mathsf{Hybrid}_{3.b}$). It then forwards the functions $G_1, \ldots, G_{\ell_{\mathrm{prg}}}$ to the challenger of CRFE. Upon receiving $\{\mathsf{CRFE.sk}_i\}_{i \in [\ell_{\mathrm{prg}}]}$ from the challenger, $\mathcal{B}$ then sends $\mathsf{CFE.sk}_f = \{\mathsf{CRFE.sk}_i\}_{i \in [\ell_{\mathrm{prg}}]}$ to the adversary.

We have to first argue that $\mathcal{B}$ is a valid adversary in the game of CRFE. To do this we argue that for every query $G_i = \mathsf{Encode}[P_i, \mathsf{TAG}_i, I_i, CT_i]$ submitted by $\mathcal{B}$, it holds that $G_i(\widetilde{y_0}) = G_i(\widetilde{y_1})$. This is because we encrypt the output $G_i(\widetilde{y_0})$ in CT_i and the output of $G_i(\widetilde{y_1})$ is the decryption of CT_i. Now that we have shown that $\mathcal{B}$ is a valid adversary, observe that if the challenger of CRFE uses $\widetilde{y_0}$ to encrypt the challenge message then we are in $\mathsf{Hybrid}_{2.b}$ and if it uses $\widetilde{y_1}$ to encrypt the challenge message then we are in $\mathsf{Hybrid}_{3.b}$. Thus, the advantage of $\mathcal{B}$ in breaking the security of CRFE is $|\mathsf{Adv}_{\mathcal{A},2.b} - \mathsf{Adv}_{\mathcal{A},3.b}|$ which, by the security of CRFE, is negligible in λ.

Proof of Lemma 3. We construct a reduction $\mathcal{B}$ that uses $\mathcal{A}$ to break the security of $\mathcal{PRF}$.

$\mathcal{B}$ answers the challenge message query as in the previous hybrid. Upon receiving a function query f, it first computes $\mathsf{Decomp}(f)$ as in $\mathsf{Hybrid}_{3.b}$ (or $\mathsf{Hybrid}_{4.b}$). Let ℓ_{prg} be the number of program components. The reduction then picks tags $\mathsf{tag}_1, \ldots, \mathsf{tag}_{\ell_R}$ at random from $\{0,1\}^\lambda$, where ℓ_R is a polynomial in $(\ell_{\mathrm{prg}}, \lambda)$. It then queries the PRF oracle to get the evaluations of tag_i for every $i \in [\ell_R]$. Denote by $\mathbf{r}$ the concatenation of the bits obtained from the evaluations. Then, $\mathcal{B}$ proceeds as in $\mathsf{Hybrid}_{3.b}$ or $\mathsf{Hybrid}_{4.b}$.

If the PRF oracle returned PRF evaluations then we are in $\mathsf{Hybrid}_{3.b}$ otherwise we are in $\mathsf{Hybrid}_{4.b}$. Thus, the advantage of $\mathcal{B}$ breaking the security of $\mathcal{PRF}$ is $|\mathsf{Adv}_{\mathcal{A},3.b} - \mathsf{Adv}_{\mathcal{A},4.b}|$ which by the security of $\mathcal{PRF}$ is negligible in λ.

Proof of Lemma 4. We construct a reduction $\mathcal{B}$ that uses $\mathcal{A}$ to break the security of PDRE. We just focus on the case when $\mathcal{A}$ makes a single message and function query. The general case when $\mathcal{A}$ makes multiple message and function queries follows from a standard hybrid argument.

$\mathcal{B}$ handles the challenge message query (x_0, x_1) as in $\mathsf{Hybrid}_{4.0}$ or $\mathsf{Hybrid}_{4.1}$. Upon receiving a function query f from $\mathcal{A}$, it forwards this along with (x_0, x_1) to the challenger of PDRE. In return it receives $(\mathsf{P}, \mathcal{S}, \mathcal{I})$ and encodings $\{\widehat{P_i, \mathbf{x}}\}_{i \in [\ell_{\mathrm{prg}}]}$. It then uses these encodings and the set systems to generate the functional key $\mathsf{CFE.sk}_f = \{\mathsf{CRFE.sk}_i\}_{i \in [\ell_{\mathrm{prg}}]}$. This functional key is then forwarded to the adversary.

We first remark that $\mathcal{B}$ is a valid adversary in the security game of PDRE. To show this, it suffices to argue that $f(x_0) = f(x_1)$. This plainly follows from the fact that $\mathcal{A}$ is a valid adversary in the game of CFE.

If the challenger of PDRE hands $\mathcal{B}$ the encodings of $\{\widehat{P_i, \mathbf{x}}\}_{i \in [\ell_{\mathrm{prg}}]}$ where $\mathbf{x} = x_0$ then we are in $\mathsf{Hybrid}_{4.0}$, else if $\mathbf{x} = x_1$ we are in $\mathsf{Hybrid}_{4.1}$. This translates to the fact that $\mathcal{B}$ wins the security game of PDRE with advantage $|\mathsf{Adv}_{\mathcal{A},4.0} - \mathsf{Adv}_{\mathcal{A},4.1}|$ which, by the security of PDRE is negligible in λ.

4 Instantiations of Program Decomposable RE

We describe two different instantiations of PD-RE: one based on based on Kilian's RE for polynomial-size branching programs [Kil88, Bar86] and another based on Yao's garbled circuits technique [Yao86]. The former instantiation helps us identify a simple function family $\mathcal{F}_{\mathrm{simple}}$ such that collusion-resistant FE for $\mathcal{F}_{\mathrm{simple}}$ suffices for our transformation in Section 3.

NC^1 **randomized encodings.** We show how to instantiate program-decomposable RE used in our transformation in Section 3 with a variant of Kilian's RE [Kil88]. Since Kilian's RE is described for polynomial-size branching programs, we start by first briefly recalling the notion of branching programs.

Kilian's RE. We work over the symmetric group S_5. A branching program with input length $\ell_{\rm inp}$ over S_5 is represented by $\mathsf{BP} = (BP, \chi)$, where $BP = \Big((g_1^0, g_1^1), \ldots, (g_{\ell_{\rm prg}}^0, g_{\ell_{\rm prg}}^1)\Big)$ and $\chi : [\ell_{\rm prg}] \to [\ell_{\rm inp}]$. Here, $g_i^b \in S_5$. The evaluation of BP on input x is $\prod_{i=1}^{\ell_{\rm prg}} g_i^{x_{\chi(i)}}$.

A randomized encoding of (BP, x) is just $\left\{r_i g_i^{x_{\chi(i)}} r_{i+1}^{-1}\right\}_{i \in [\ell_{\rm prg}]}$, where $r_1, r_{\ell_{\rm prg}+1} = \mathbf{1}_{S_5}$ (the identity in S_5 is denoted by $\mathbf{1}_{S_5}$) and r_i, $\forall i \in [2, \ell_{\rm prg}]$, is sampled at random from S_5. To evaluate, just compute the product of all the group elements in the encoding. That is, the evaluation is $\prod_{i=1}^{\ell_{\rm prg}} r_i g_i^{x_{\chi(i)}} r_{i+1}^{-1}$. Note that this is same as evaluating BP on x.

Program-decomposable RE. We now describe how to derive a PD-RE scheme from Kilian's RE. We refer to the resulting PD-RE scheme as $\mathsf{PDRE}_{\mathsf{BP}}$.

- Decomp: It takes as input a branching program $\mathsf{BP} = (BP, \chi)$, where BP and χ are as defined above. It then assigns the program component $P_i = (i, g_i^0, g_i^1)$. The final program is $\mathsf{P} = \{P_i\}_{i \in [\ell_{\rm prg}]}$. The set systems are computed in the following manner: first, construct a string r of length ℓ_R. This string comprises of blocks with $\mathsf{Block}_1 = (\perp, r_2)$, $\mathsf{Block}_{\ell_{\rm prg}} = (r_{\ell_{\rm prg}}, \perp)$ and $\mathsf{Block}_i = (r_i, r_{i+1})$ for $i \in [2, \ell_{\rm prg} - 1]$. The set S_i is set to be all the positions in r corresponding to Block_i. The set system $\mathcal{S}$ is set to $\{S_i\}_{i \in [\ell_{\rm prg}]}$. Similarly, the set I_i is set to be $\{\chi(i)\}$. And the set system $\mathcal{I}$ is just $\{I_i\}_{i \in [\ell_{\rm prg}]}$. The final output is $(\mathsf{P}, \mathcal{S}, \mathcal{I}, \ell_R)$.

- PrgEnc: It takes as input $\Big(P_i = (i, g_i^0, g_i^1), x_{\chi(i)}; \mathbf{r}_i = r_{|S_i}\Big)$. It then parses $\mathbf{r}_i$ as (u, v). There are three cases:

  - Case 1. $i = 1$: It computes $\tilde{g} = g_i^{x_{\chi(i)}} \cdot v^{-1}$. It outputs $\widehat{P_i, x} = \tilde{g}$.
  - Case 2. $i = \ell_{\rm prg}$: It computes $\tilde{g} = u \cdot g_i^{x_{\chi(i)}}$. It outputs $\widehat{P_i, x} = \tilde{g}$.
  - Case 3. $i \in [2, \ell_{\rm prg} - 1]$: It computes $\tilde{g} = u \cdot g_i^{x_{\chi(i)}} \cdot v^{-1}$. It outputs $\widehat{P_i, x} = \tilde{g}$.

  This completes the description of PrgEnc.

- Dec: It takes as input $\left(\widehat{P_i, x}\right)_{i \in [\ell_{\rm prg}]}$ and outputs $\prod_{i=1}^{\ell_{\rm prg}} \widehat{P_i, x}$.

The output distribution of the program encoding algorithm is identical to the output distribution of Killian's RE.

We remark that the above PD-RE scheme can only be defined for NC^1 circuits since the existence of poly-sized branching programs for $P/poly$ is not known.

Compact FE from Collusion-Resistant FE for "Simple" Functions. We now describe a concrete function family $\mathcal{F}_{\rm simple}$ that suffices for our transformation in Section 3.

Let BP be a polynomial-size branching program. Let $\mathsf{BP} = (BP, \chi)$, where $BP = ((g_1^0, g_1^1), \ldots, (g_{\ell_{\rm prg}}^0, g_{\ell_{\rm prg}}^1))$. With respect to every element in the branching program BP, we define a 120×120 table ($\because |S_5| = 120$).

The Table *function.* We now define a function Table as follows:

- Case 1. $i = 1$: For $b \in \{0,1\}$, we have $\mathsf{Table}_{g_i^b}[u][v] = g_i^b \cdot v^{-1}$.
- Case 2. $i = \ell_{\text{prg}}$: For $b \in \{0,1\}$, we have $\mathsf{Table}_{g_i^b}[u][v] = u \cdot g_i^b$.
- Case 3. $i \in [2, \ell_{\text{prg}} - 1]$: For $b \in \{0,1\}$, we have $\mathsf{Table}_{g_i^b}[u][v] = u \cdot g_i^b \cdot v^{-1}$.

Function family $\mathcal{F}_{\text{simple}}$. Given, the function Table, we can now replace the generic function family $\mathcal{F}_{\mathsf{CRFE}}$ (consisting of functions Encode) used in our transformation in Section 3 with a simple function family $\mathcal{F}_{\text{simple}}$ consisting of functions f_{simple} described in Figure 3. We replace the generic encryption scheme used in Encode with the concrete symmetric encryption scheme defined in [Gol09], where an encryption of a message m is $(s, PRF_K(s) \oplus m)$.

$\underline{f_{\text{simple}}[P = (i, g_i^0, g_i^1), \tau_i, \tau_{i+1}, I, CT = (s_0, s_1)](x, K, \mathsf{Sym}.K, \mathsf{mode})}$

If $\mathsf{mode} = 0$:

- $u \leftarrow PRF_K(\tau_i)$.
- $v \leftarrow PRF_K(\tau_{i+1})$.
- If $x_{\chi(i)} = 0$, output $\mathsf{Table}_{g_i^0}[u][v]$. Else, output $\mathsf{Table}_{g_i^1}[u][v]$.

If $\mathsf{mode} = 1$:

- Output $PRF_K(s_0) \oplus s_1$.

Figure 3

By making the above modifications in our transformation, we obtain the following theorem:

Theorem 4 *Let $\mathcal{F}_{\text{simple}}$ be the function family described in Figure 3. A public-key FE scheme for $\mathcal{F}_{\text{simple}}$ with selective-security against unbounded collusions implies a compact FE scheme for NC^1 with selective-security.*

Garbled circuits. We show that Yao's garbled circuits [Yao86] can be viewed as an instantiation of the Program Decomposable RE scheme we define in Section 3. Before we show this, we briefly recall the notion of garbled circuits. We first define an encoder that takes as input a circuit C and input x. In the following description, we assume that C has fan-in 2 and fan-out 1. The encoder then computes an encoding of $C(x)$ as follows. First, two wire keys are associated to every wire denoting bits 0 and 1.

1. A table of ciphertexts is then created for every gate in the circuit. To be more precise, there is a PPT algorithm $\mathsf{GateGarb}$ that takes as input a description of gate G, keys (K_0^a, K_1^a) corresponding to its first input wire w_a, keys (K_0^b, K_1^b) corresponding to its first second wire w_b and keys (K_0^c, K_1^c) corresponding to its output wire w_c of G. It then outputs a table T_G. We denote the randomness taken by $\mathsf{GateGarb}$ to be r_G. It is important to note here that $|r_G|$ depends only on λ (and not on the size of C).
2. There is a deterministic procedure $\mathsf{ChooseInp}$ that chooses one of the input wire keys corresponding to the appropriate bit of x. That is, $\mathsf{ChooseInp}$ takes as input W_i, the description of i^{th} input wire of C, along with its associated keys (K_0^i, K_1^i) and it outputs the key $K_{x_i}^i$.

So the final garbled circuit is $\left(\{T_G\}_{G \in \mathsf{Gates}(C)}, \{K_{x_i}^i\}_{i \in [|x|]}\right)$, where $\mathsf{Gates}(C)$ is a set of all gates in C.

We now show to build an Program Decomposable RE scheme using garbled circuits.

- $\mathsf{Decomp}(C)$: It takes as input a circuit C. The program components are nothing but the gates in C and its input wires. That is, $\mathsf{P} = \left\{ \{P_G = G\}_{G \in \mathsf{Gates}(C)}, \{P_i\}_{i \in [\ell_{\mathrm{inp}}]} \right\}$, where P_i is the description of i^{th} input wire of C and ℓ_{inp} is the input length of C. We now deal with computing the set systems. We first construct a string r. This string is made up of blocks, one for every gate and every input wire of C. A block corresponding to a gate G, denoted by Block_G, contains the wire keys of G's input wires and its output wire. Furthermore, Block_G also contains r_G which is the randomness used by $\mathsf{GateGarb}$ to compute the table of ciphertexts corresponding to G. We note that two blocks corresponding to two different gates could contain same strings. For example, let the output wire of G be fed to the gate G' through the wire w. Then, both Block_G and $\mathsf{Block}_{G'}$ contain wire keys of w. A block corresponding to the i^{th} input wire key w_i of G, denoted by Block_i, contains the wire keys corresponding to w_i.

  Let $\ell_R = |r|$. The set $S_G \subseteq [\ell_R]$ comprises of all the positions in r corresponding to Block_G -substring of r. Further the set S_i, associated to the i^{th} input wire of C, comprises of all the positions in r corresponding to Block_i -substring of r. It then computes the set system $\mathcal{S} = \left\{ \{S_G\}_{G \in \mathsf{Gates}(C)}, \{S_i\}_{i \in [\ell_{\mathrm{inp}}]} \right\}$. On the other hand, $I_i = \{i\}$ for all $i \in [\ell_{\mathrm{inp}}]$. The set system $\mathcal{I}$ is then set to be $\{I_i\}_{i \in [\ell_{\mathrm{inp}}]}$.

  Finally, it outputs $\left(\{P_G\}_{G \in \mathsf{Gates}(C)}, \mathcal{S}, \mathcal{I}, \ell_R \right)$.

- $\mathsf{PrgEnc}(P_i, x_i; r_i = r_{|S_i})$: It takes as input program component P_i, input bit x_i and randomness r_i. If P_i is a gate, it does the following. It parses r_i as a sequence of wire keys, denoted by $\mathbf{K}$, and string r_G. It then executes $\mathsf{GateGarb}(P_i, \mathbf{K}; r_G)$ and the resulting T_G is output. Else if P_i is an input wire key, it executes $\mathsf{ChooseInp}(P_i, r_i)$ to obtain $K^i_{x_i}$ which is then output.

- $\mathsf{Dec}\left(\{T_G\}_{G \in \mathsf{Gates}(C)}, \{K^i_{x_i}\}_{i \in [\ell_{\mathrm{inp}}]} \right)$: It takes as input table of ciphertexts w.r.t to every gate G in the circuit and input wire keys $K^i_{x_i}$. It then executes the garbled circuit decoding procedure to recover the output y.

5 Implications to $i\mathcal{O}$

Here we state the implications of our main result towards achieving general-purpose $i\mathcal{O}$.

$i\mathcal{O}$ from Collusion-Resistant FE. We first recall the main result of [AJ15, BV15a]:

Theorem 5 ([AJ15, BV15a]) *Public-key compact FE for NC^1 with sub-exponential security in the selective model for a single key query implies i$\mathcal{O}$ for P/Poly.*

Combining Theorem 5 with our transformation from Section 3, we obtain the following:

Theorem 6 *Public-key FE for NC^1 with sub-exponential security in the selective model against unbounded collusions implies i$\mathcal{O}$ for P/Poly.*

Combining Theorem 4 in Section 4 with Theorem 5, we obtain:

Theorem 7 *Public-key FE for $\mathcal{F}_{\mathrm{simple}}$ (see Figure 3) with sub-exponential security in the selective model against unbounded collusions implies i$\mathcal{O}$ for P/Poly.*

References

[AB15] Benny Applebaum and Zvika Brakerski. Obfuscating circuits via composite-order graded encoding. In *TCC*, 2015.

[ABSV15] Prabhanjan Ananth, Zvika Brakerski, Gil Segev, and Vinod Vaikuntanathan. From selective to adaptive security in functional encryption. In *CRYPTO*, 2015.

[AGIS14] Prabhanjan Ananth, Divya Gupta, Yuval Ishai, and Amit Sahai. Optimizing obfuscation: Avoiding barrington's theorem. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 646–658. ACM, 2014.

[AIK04] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Cryptography in nc^0. In *45th Symposium on Foundations of Computer Science (FOCS 2004), 17-19 October 2004, Rome, Italy, Proceedings*, pages 166–175, 2004.

[AIK06] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Computationally private randomizing polynomials and their applications. *Computational Complexity*, 15(2):115–162, 2006.

[AJ15] Prabhanjan Ananth and Abhishek Jain. Indistinguishability obfuscation from compact functional encryption. In *CRYPTO*, 2015.

[App11] Benny Applebaum. Randomly encoding functions: A new cryptographic paradigm - (invited talk). In *Information Theoretic Security - 5th International Conference, ICITS 2011, Amsterdam, The Netherlands, May 21-24, 2011. Proceedings*, pages 25–31, 2011.

[App14] Benny Applebaum. Bootstrapping obfuscators via fast pseudorandom functions. In *Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014, Proceedings, Part II*, pages 162–172, 2014.

[Bar86] David A. Mix Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in nc^1. In *Proceedings of the 18th Annual ACM Symposium on Theory of Computing, May 28-30, 1986, Berkeley, California, USA*, pages 1–5, 1986.

[BCG+11] Nir Bitansky, Ran Canetti, Shafi Goldwasser, Shai Halevi, Yael Tauman Kalai, and Guy N. Rothblum. Program obfuscation with leaky hardware. In *Advances in Cryptology - ASIACRYPT 2011 - 17th International Conference on the Theory and Application of Cryptology and Information Security, Seoul, South Korea, December 4-8, 2011. Proceedings*, pages 722–739, 2011.

[BGI+01] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In Joe Kilian, editor, *Advances in Cryptology - CRYPTO 2001, 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19-23, 2001, Proceedings*, volume 2139 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2001.

[BGJ+15] Nir Bitansky, Shafi Goldwasser, Abhishek Jain, Omer Paneth, Vinod Vaikuntanathan, and Brent Waters. Time-lock puzzles from randomized encodings. *IACR Cryptology ePrint Archive*, 2015:514, 2015.

[BGK+14] Boaz Barak, Sanjam Garg, Yael Tauman Kalai, Omer Paneth, and Amit Sahai. Protecting obfuscation against algebraic attacks. In *Advances in Cryptology - EUROCRYPT 2014 - 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings*, pages 221–238, 2014.

[BGL+15] Nir Bitansky, Sanjam Garg, Huijia Lin, Rafael Pass, and Siddartha Telang. Succinct randomized encodings and their applications. In *STOC*, 2015.

[BMR90] Donald Beaver, Silvio Micali, and Phillip Rogaway. The round complexity of secure protocols (extended abstract). In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing, May 13-17, 1990, Baltimore, Maryland, USA*, pages 503–513, 1990.

[BR14] Zvika Brakerski and Guy N. Rothblum. Virtual black-box obfuscation for all circuits via generic graded encoding. In *TCC*, pages 1–25, 2014.

[BS15] Zvika Brakerski and Gil Segev. Function-private functional encryption in the private-key setting. In *TCC*, 2015.

[BSW11] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In *Theory of Cryptography*, pages 253–273. Springer, 2011.

[BV15a] Nir Bitansky and Vinod Vaikuntanathan. Indistinguishability obfuscation from functional encryption. In *FOCS*, 2015.

[BV15b] Nir Bitansky and Vinod Vaikuntanathan. Indistinguishability obfuscation from functional encryption. Cryptology ePrint Archive, Report 2015/163, 2015. http://eprint.iacr.org/.

[BWZ14] Dan Boneh, David J. Wu, and Joe Zimmerman. Immunizing multilinear maps against zeroizing attacks. *IACR Cryptology ePrint Archive*, 2014:930, 2014.

[CGH+15] Jean-Sébastien Coron, Craig Gentry, Shai Halevi, Tancrède Lepoint, Hemanta K. Maji, Eric Miles, Mariana Raykova, Amit Sahai, and Mehdi Tibouchi. Zeroizing without low-level zeroes: New MMAP attacks and their limitations. In *CRYPTO*, 2015.

[CHL+15] Jung Hee Cheon, Kyoohyung Han, Changmin Lee, Hansol Ryu, and Damien Stehlé. Cryptanalysis of the multilinear map over the integers. In *EUROCRYPT*, 2015.

[CHV15] Aloni Cohen, Justin Holmgren, and Vinod Vaikuntanathan. Publicly verifiable software watermarking. *IACR Cryptology ePrint Archive*, 2015:373, 2015.

[CIJ+13] Angelo De Caro, Vincenzo Iovino, Abhishek Jain, Adam O'Neill, Omer Paneth, and Giuseppe Persiano. On the achievability of simulation-based security for functional encryption. In *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part II*, pages 519–535, 2013.

[CLT13] Jean-Sébastien Coron, Tancrède Lepoint, and Mehdi Tibouchi. Practical multilinear maps over the integers. In *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I*, pages 476–493, 2013.

[CLT14] Jean-Sébastien Coron, Tancrède Lepoint, and Mehdi Tibouchi. Cryptanalysis of two candidate fixes of multilinear maps over the integers. *IACR Cryptology ePrint Archive*, 2014:975, 2014.

[CLT15] Jean-Sébastien Coron, Tancrède Lepoint, and Mehdi Tibouchi. New multilinear maps over the integers. In *CRYPTO*, 2015.

[CLTV15] Ran Canetti, Huijia Lin, Stefano Tessaro, and Vinod Vaikuntanathan. Obfuscation of probabilistic circuits and applications. In *Theory of Cryptography - 12th Theory of Cryptography Conference, TCC 2015, Warsaw, Poland, March 23-25, 2015, Proceedings, Part II*, pages 468–497, 2015.

[GGH13a] Sanjam Garg, Craig Gentry, and Shai Halevi. Candidate multilinear maps from ideal lattices. In Thomas Johansson and Phong Q. Nguyen, editors, *Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings*, volume 7881 of *Lecture Notes in Computer Science*, pages 1–17. Springer, 2013.

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

[GGH15] Craig Gentry, Sergey Gorbunov, and Shai Halevi. Graph-induced multilinear maps from lattices. In *Theory of Cryptography - 12th Theory of Cryptography Conference, TCC 2015, Warsaw, Poland, March 23-25, 2015, Proceedings, Part II*, pages 498–527, 2015.

[GGHR14] Sanjam Garg, Craig Gentry, Shai Halevi, and Mariana Raykova. Two-round secure MPC from indistinguishability obfuscation. In *Theory of Cryptography - 11th Theory of Cryptography Conference, TCC 2014, San Diego, CA, USA, February 24-26, 2014. Proceedings*, pages 74–94, 2014.

[GGHZ14] Sanjam Garg, Craig Gentry, Shai Halevi, and Mark Zhandry. Fully secure functional encryption without obfuscation. *IACR Cryptology ePrint Archive*, 2014:666, 2014.

[GHMS14] Craig Gentry, Shai Halevi, Hemanta K. Maji, and Amit Sahai. Zeroizing without zeroes: Cryptanalyzing multilinear maps without encodings of zero. *IACR Cryptology ePrint Archive*, 2014:929, 2014.

[GIS+10] Vipul Goyal, Yuval Ishai, Amit Sahai, Ramarathnam Venkatesan, and Akshay Wadia. Founding cryptography on tamper-proof hardware tokens. In *Theory of Cryptography, 7th Theory of Cryptography Conference, TCC 2010, Zurich, Switzerland, February 9-11, 2010. Proceedings*, pages 308–326, 2010.

[GJKS15] Vipul Goyal, Abhishek Jain, Venkata Koppula, and Amit Sahai. Functional encryption for randomized functionalities. In *Theory of Cryptography - 12th Theory of Cryptography Conference, TCC 2015, Warsaw, Poland, March 23-25, 2015, Proceedings, Part II*, pages 325–351, 2015.

[GKP+13] Shafi Goldwasser, Yael Tauman Kalai, Raluca A. Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich. Reusable garbled circuits and succinct functional encryption. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 555–564. ACM, 2013.

[GLSW15] Craig Gentry, Allison B. Lewko, Amit Sahai, and Brent Waters. Indistinguishability obfuscation from the multilinear subgroup elimination assumption. In *FOCS*, 2015.

[Gol09] Oded Goldreich. *Foundations of Cryptography: Volume 2, Basic Applications*, volume 2. Cambridge university press, 2009.

[GVW12] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Functional encryption with bounded collusions via multi-party computation. In *Advances in Cryptology - CRYPTO 2012 - 32nd Annual Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2012. Proceedings*, pages 162–179, 2012.

[Had00] Satoshi Hada. Zero-knowledge and code obfuscation. In *Advances in Cryptology - ASIACRYPT 2000, 6th International Conference on the Theory and Application of Cryptology and Information Security, Kyoto, Japan, December 3-7, 2000, Proceedings*, pages 443–457, 2000.

[IK00] Yuval Ishai and Eyal Kushilevitz. Randomizing polynomials: A new representation with applications to round-efficient secure computation. In *41st Annual Symposium on Foundations of Computer Science, FOCS 2000, 12-14 November 2000, Redondo Beach, California, USA*, pages 294–304, 2000.

[Kil88] Joe Kilian. Founding cryptography on oblivious transfer. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA*, pages 20–31, 1988.

[KSY15] Ilan Komargodski, Gil Segev, and Eylon Yogev. Functional encryption for randomized functionalities in the private-key setting from minimal assumptions. In *Theory of Cryptography - 12th Theory of Cryptography Conference, TCC 2015, Warsaw, Poland, March 23-25, 2015, Proceedings, Part II*, pages 352–377, 2015.

[NW15] Ryo Nishimaki and Daniel Wichs. Watermarking cryptographic programs against arbitrary removal strategies. *IACR Cryptology ePrint Archive*, 2015:344, 2015.

[O'N10] Adam O'Neill. Definitional issues in functional encryption. *IACR Cryptology ePrint Archive*, 2010:556, 2010.

[PST14] Rafael Pass, Karn Seth, and Sidharth Telang. Indistinguishability obfuscation from semantically-secure multilinear encodings. In *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part I*, pages 500–517, 2014.

[SS10] Amit Sahai and Hakan Seyalioglu. Worry-free encryption: functional encryption with public keys. In *Proceedings of the 17th ACM Conference on Computer and Communications Security, CCS 2010, Chicago, Illinois, USA, October 4-8, 2010*, pages 463–472, 2010.

[SW05] Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In *Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005, Proceedings*, pages 457–473, 2005.

[SW14] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In David B. Shmoys, editor, *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 475–484. ACM, 2014.

[SZ14] Amit Sahai and Mark Zhandry. Obfuscating low-rank matrix branching programs. Technical report, Cryptology ePrint Archive, Report 2014/773, 2014. http://eprint. iacr. org, 2014.

[Wat14] Brent Waters. A punctured programming approach to adaptively secure functional encryption. In *CRYPTO*, 2014.

[Yao86] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *FOCS*, pages 162–167, 1986.

[Zim15] Joe Zimmerman. How to obfuscate programs directly. In *EUROCRYPT*, 2015.

A Bootstrapping Theorem for iO

In this section, we prove a bootstrapping theorem for iO. Concretely, we show how to construct FE for $\mathcal{F}_{\mathrm{simple}}$ from iO for function family $\mathcal{F}$. This, when combined with our main result yields iO for arbitrary circuits.

Prior works [GGH+13b, Wat14] present constructions of FE based on iO and additional well studied cryptographic assumptions. Of particular interest is the work of [Wat14] who show how to build FE based on iO and one-way functions. Using this work, we get a bootstrapping theorem for iO: from [Wat14], we have FE for $\mathcal{F}_{\mathrm{simple}}$ from iO for $\mathcal{F}$ (where $\mathcal{F}$ is described next) and then using the result on iO for P/poly from FE for $\mathcal{F}_{\mathrm{simple}}$, we achieve iO for P/poly from iO for $\mathcal{F}$.

We describe the function family $\mathcal{F}$ in Figure 4 that is sufficient to obtain FE for $\mathcal{F}_{\mathrm{simple}}$ via [Wat14]. In [Wat14], there are different functions that are obfuscated as part of the public key and the functional key whose description also change in the hybrids. We take all this into account while designing the template for all functions in $\mathcal{F}_{\mathrm{simple}}$. Furthermore, in the description of the obfuscated program as part of the functional key for f, we substitute f with the concrete functions in $\mathcal{F}_{\mathrm{simple}}$.

We formally state the bootstrapping theorem below.

Theorem 8 *For any circuit class $\mathcal{C} = \{C : \{0,1\}^n \to \{0,1\}\}$, there exists an iO scheme for $\mathcal{C}$ assuming the existence of sub exponentially secure iO for $\mathcal{F}$ and sub exponentially secure one one-way functions. Further, $|f| = \mathrm{poly}(\lambda, n)$, for every $f \in \mathcal{F}$.*

$f \in \mathcal{F}$

Hardwired value: v, (puncturable) PRF key K.

1. Execute one of the following steps:
   - $\perp$
   - PRG evaluation
   - Simple IF-ELSE check based on v and the input.
2. Puncturable PRF evaluation.
3. Execute one of the following steps:
   - $\perp$
   - Two levels of successive PRF evaluations, followed by constant operations.
     Remark: In Waters [Wat14], the last step of the program in the functional key for f, is the evaluation of f on the decryption of a ciphertext. The decryption ciphertext is one level of PRF evaluation. Also, $f \in \mathcal{F}_{\text{simple}}$ also comprises of PRF evaluations followed by constant operations. This is the second level of PRF evaluations.

Figure 4 Every function in the function family $\mathcal{F}$ has the above template. $\perp$ indicates that this particular step is empty.

A consequence of the above theorem is that for every (sufficiently large) input length $n \in \mathbb{N}$, in order to obfuscate $C \in P/poly$ it suffices to obfuscate a class of functions whose size is a *fixed* polynomial in the security parameter for n. And in particular, only depends on the input length of circuit C.