Indistinguishability Obfuscation from Bilinear Maps and Block-Wise Local PRGs

Huijia Lin* Stefano Tessaro †

University of California, Santa Barbara
{rachel.lin,tessaro}@cs.ucsb.edu

Abstract

Recent works (Lin, EUROCRYPT'16, ePrint'16; Lin and Vaikunthanathan, FOCS'16; Ananth and Sahai, EUROCRYPT'17) establish a tight connection between constructions of indistinguishability obfuscation from L-linear maps and pseudo-random generators (PRGs) with output locality L. This approach appears however not to be suitable to obtain instantiations from bilinear maps, as no polynomial-stretch PRG with locality lower than 5 exists.

This paper presents new candidate constructions of indistinguishability obfuscation from *(i)* L-linear maps for any $L \geq 2$, and *(ii)* PRGs with *block-wise* locality L. A PRG has block-wise locality L if every output bit depends on at most L (disjoint) *input blocks*, each consisting of up to $\log \lambda$ input bits. In particular, we give:

- A construction of a general-purpose indistinguishability obfuscator from L-linear maps and a subexponentially-secure PRG with block-wise locality L and polynomial stretch.
- A construction of general-purpose functional encryption from L-linear maps and any slightly super-polynomially secure PRG with block-wise locality L and polynomial stretch.

All our constructions are based on the SXDH assumption on L-linear maps and subexponential Learning With Errors (LWE) assumption. In the special case of $L = 2$, our constructions can be alternatively based on bilinear maps with the Matrix Diffie-Hellman assumption and the 3-party Decision Diffie Hellman assumption, without assuming LWE.

Concurrently, we initiate the study of candidate PRGs with block-wise locality $L \geq 2$ based on Goldreich's local functions, and their security. In particular, lower bounds on the locality of PRGs do not apply to block-wise locality for any $L \geq 2$, and the security of instantiations with block-wise locality $L \geq 3$ is backed by similar validation as constructions with (conventional) locality 5. We complement this with hardness amplification techniques that weaken the pseudorandomness requirement on our candidates to qualitatively weaker requirements.

*Huijia Lin was partially supported in part by NSF grants CNS-1528178 and CNS-1514526.

†Research supported in part by NSF grants CNS-1423566, CNS-1528178, CNS-1553758 (CAREER), and IIS-152804.

Contents

1 Introduction

Indistinguishability obfuscation (IO), first defined in the seminal work of Barak *et al.* [BGI$^+$01a], aims to obfuscate functionally equivalent programs into indistinguishable ones while preserving functionality. IO is an extraordinarily powerful object that has been shown to enable a whole set of new cryptographic possibilities. Currently, all IO constructions [GGH$^+$13b, BR14, BGK$^+$14, PST14, AGIS14, GLSW15, Zim15, AB15, GMS16, MSZ16a, Lin16a, LV16, Lin16b, AS16] rely on *multilinear maps* or *graded encodings*. In particular, the power of an L-linear map stems from the fact that it essentially allows to evaluate degree-L polynomials on secret encoded values, and to test whether the output of such polynomials is zero or not.

While the case $L = 2$ corresponds to bilinear maps, which can be efficiently instantiated from elliptic curves, the instantiation of L-linear maps with $L \geq 3$ appears to be a challenging problem. Garg, Gentry, and Halevi [GGH13a] proposed in particular *noisy* (i.e., *approximate*) versions of L-linear maps for $L \geq 3$, and gave the first candidate construction. Unfortunately, vulnerabilities [CHL$^+$15, CGH$^+$15, MSZ16b, CGH16] were later demonstrated against this and subsequent candidates [CLT13, LSS14, GGH15, CLT15]. These attacks have motivated efforts towards building IO from L-linear maps, where L is as small as possible. The current state-of-the-art [Lin16b, AS16] shows that 5-linear maps suffice (under appropriate assumptions), but it remains unclear whether such maps are more likely to exist than L-linear maps for a higher L.

Therefore, the obvious goal would be to provide a construction of IO from *bilinear* maps, thus completely dispensing with multilinear maps for $L \geq 3$. The challenge here is to be traced back to the clear gap between the functionality IO demands — evaluating arbitrary polynomial-sized circuits — and the functionality offered by bilinear maps — evaluating only *quadratic* polynomials. The perhaps most striking indication of this challenge is the mere fact that we do not even have a plausible correct IO construction from bilinear maps, let alone a secure one.

This paper, in a nutshell This paper presents the first IO candidate construction which relies on L-linear maps for $L \geq 2$. In particular, we obtain the *first* IO candidate construction from bilinear maps, based on well-specified assumptions we discuss below.

We fundamentally rely on the recent line of works on building IO from constant-degree multilinear maps [Lin16a, LV16, Lin16b, AS16], which all rely on so-called *local* pseudo-random generators (PRGs) – a PRG with *locality* L has every output bit depend on L input bits. It is known that if PRGs with locality L and polynomial stretch exist, then IO can be constructed from L-linear maps [Lin16b, AS16]. Unfortunately, we do not even have locality-4 (polynomial stretch) PRGs [CM01, MST03], and candidate PRGs only exist starting from locality 5 [Gol01, MST03, OW14]. To circumvent the lower bound on PRG locality, we propose a new, relaxed, notion of locality, called *block-wise locality*. We build upon Lin's [Lin16b] recent IO construction, but show that in order to obtain IO from L-linear maps, it suffices to use PRGs with block-wise locality L. As we will discuss below, such PRGs can exist for L as low as two.

Block-wise locality and IO We say that a PRG mapping $n \times \ell$ input bits to m output bits has *block-wise locality* L and *block-size* ℓ, if when viewing its input (i.e., the seed) as a matrix of $n \times \ell$ bits, every output bit depends on at most L *columns* in the matrix (as opposed to L input bits), as depicted in Figure 1. Observe that that the actual locality of such PRGs can go up to $L \times \ell$, yet, it has the special structure that all these input bits come from merely L input columns. This special structure is the key feature that allows for replacing local PRGs with block-wise-local PRGs, in the following applications.

- *Application I:* If there exists a *subexponentially-secure* PRG with block-wise-locality L, and any block-size $\ell = O(\log \lambda)$, then we can construct general-purpose IO from L-linear maps.
- *Application II:* If the block-wise local PRG is only *slighly superpolynomially secure*, we can still build special-purpose IO for circuits with super-logarithmic length inputs, which implies full-fledged Functional Encryption (FE), from L-linear maps.

All our constructions come with security reductions to (1) the security of block-wise-local PRGs, (2) the SXDH assumption on L-linear maps, with the same level of hardness as that of the PRG, and (3) the subexponential Learning With Errors (LWE). When $L = 2$, our constructions can be alternatively based on two other assumptions on bilinear maps and without assuming LWE.

Concurrently, we investigate the existence of block-wise local PRGs. We propose candidates following the common paradigm for candidate local PRGs [CM01, MST03, App12, OW14, AL16], which are variants of Goldreich's functions [Gol00]. We simply replace every PRG input bit with a column of ℓ input bits. Such a block-wise local PRG is parameterized by an bipartite expander graph and a predicate (or potentially a set of predicates) over $L \times \ell$ input bits. We discuss the security of these candidates, against known attacks, in relation to the choice of graph and predicate. Furthermore, aiming at weakening the assumption on our candidates, we present two hardness amplification techniques that amplify respectively the weaker *next-bit-unpredicatability* property and *pseudo-min-entropy generation* property to different levels of pseudorandomness guarantees.

A perspective We see our study of block-wise local PRGs as a first step towards understanding their (in)security, and we call for more extensive study, especially for the case of block-wise locality 2. Still, two important comments are in order.

First, for block-wise locality $L \geq 3$, our assumption is implied by that made by recent works in the area of local PRGs and PRFs, c.f. e.g. the pseudorandomness assumptions from the recent work by Applebaum and Raykov [AR16] — and in fact, our amplification results show that even less needs to be achieved by the local function. Second, while the case $L = 2$ requires extra care, we note that this gives at the very least the first obfuscator candidate with a clear reduction to well-defined assumption for bilinear maps.

1.1 Block-Wise Locality

A $(n \times \ell, m)$-PRG maps $n \times \ell$ input bits to m output bits. As introduced above, a PRG has block-wise locality L and block-size ℓ, if when viewing the input as a $n \times \ell$ matrix, every output bits depend on input bits in at most L columns. Such a function is fully specified by the input-output dependency graph G describing which input columns each output bit depends on, and the set of predicates $\{P_j\}_{j \in [m]}$ that each output bit is evaluated through.

In all our applications, we consider block-wise local PRGs with sufficiently large polynomial input- and output-lengths, n and m (in the security parameter λ) and logarithmic block-size $\ell = O(\log(\lambda))$. In this setting, a PRG has polynomial-stretch if $m = n^{1+\alpha}$ for some positive constant $\alpha > 0$. For convenience, below we assume such parameters are fixed in our discussion.

When compared with traditional local PRGs (which can be thought as the special case with block size $\ell = 1$), the advantage of block-wise local PRGs is that while they will still permit instantiations with L-linear maps in our applications, their output bits depend on $L \times \ell$ input bits, and hence we can use more complex, say logarithmic-degree, predicates. For this reason, known lower bounds on the locality of PRGs do not apply to block-wise locality, even when $L = 2$, when the block size satisfies $\ell = \Omega(\log(\lambda))$. Observe that inverting functions with block-wise locality 2

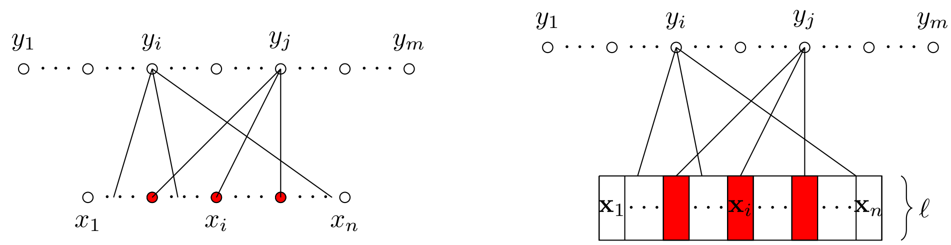

Figure 1: Left: PRG with locality $L = 3$. Right: PRG with block-wise locality $L = 3$ and block size ℓ.

corresponds to Constraint Satisfaction Problems (CSPs) where every constraint depends on two input variables from a polynomial-sized alphabet. Such CSPs are not known to be solvable in polynomial time. In constrast, if $\ell = 1$, then the problem collapses to a 2SAT instance, which is easily solvable. Moreover, the lower bounds in [CM01, MST03] show that for conventional locality, PRGs with polynomial stretch require $L \geq 5$, but they crucially rely on the fact that any locality-4 predicate is correlated with two of its input bits to rule out the existence of locality-4 PRGs. In contrast, a PRG with block-wise locality 2 can use predicates that depend on $2 \log \lambda$ input bits; setting the predicate to be uncorrelated with any subset of $\log \lambda$ input bits circumvents the lower bound argument in [CM01, MST03].

Block-wise local PRGs via local PRGs Every function with block-wise locality L and block size ℓ is a function with locality $L\ell$. Therefore, the rich literature on the security of Goldreich's local functions (see Applebaum's survey [App15]) provides guidelines on how to choose candidate block-wise local PRGs, more specifically, the dependency graph G and predicates $\{P_j\}$. In particular, the graph G should be (k, c)-expanding, i.e., every subset of $k' \leq k$ output bits depends on at least $c \times k'$ input columns, for appropriately large k and c. We show that for $L \geq 3$, a large $1 - o(1)$ fraction of graphs G is $(n^{1-\eta}, (1-\eta)L)$-expanding. This in turn means that we can think of this as an instance of Goldreich's function with locality $L\ell$ built from a graph which is $(n^{1-\eta}, (1-\eta)L\ell)$-expanding, thus taking us back to the classical setting studied in the literature.

Using this analogy, we can show for example that for block-wise locality 3 and block size 2, for most graphs G, the resulting function withstands all linear attacks with sub-exponential bias ϵ when using the predicate outputting $x_1^0 \oplus x_2^0 \oplus x_3^0 \oplus (x_1^1 \wedge x_2^1)$ on input three columns $(x_1^0, x_1^1), (x_2^0, x_2^1), (x_3^0, x_3^1)$. This is a criterion that has been adopted so far to validate PRG security of local functions.

Moving even one step further, Applebaum and Raykov [AR16] recently postulated the following (even stronger) pseudorandomness assumption on functions with logarithmic locality:

Assumption 1 (Informal). *For locality $D = O(\log \lambda)$, and arbitrarily polynomial output length $m = n^{1+\alpha}$, there exist a suitable predicate, P', such that, for* any *dependency graph G' that is $(n^{1-\eta}, (1-\eta)D)$-expanding for some $0 < \eta < 1/2$, the locality-D function specified by P' and G' is $2^{-n^{1-\eta}}$-pseudorandom again $2^{n^{1-\eta}}$-time distinguishers.*

In our setting, for block-wise locality $L \geq 3$ and block-size $\log \lambda$, we show that when choosing the dependency graph G at random, the obtained block-wise local function can be thought as a function with locality $D = L \log \lambda$ satisfying the properties specified by the Applebaum-Raykov assumption, with $1 - o(1)$ probability. In particular, such functions withstand myopic inversion attacks (cf. e.g. [CEMT09]). In fact, our applications only need pseudorandomness to hold for output length $m = n^{1+\alpha}$ for *some* arbitrarily small constant $\alpha > 0$, and against polynomial time attackers, thus a much weaker requirement than what is guaranteed by the Applebaum-Raykov assumption.

For the case $L = 2$, the assumption that a block-wise local PRG exists is not backed by any of the past results. Indeed, they all fail since the resulting graphs G is not sufficiently expanding, specifically, the graphs are $(n^{1-\eta}, \beta D)$-expanding with $D = 2 \log \lambda$, for some expansion factor $\beta < 1/2$ and $\eta < 1$; however, the proofs of previous works only go through if $\beta > 1/2$. Nevertheless, this still implies something highly non-trivial, namely that each set of output bits depends on a sufficiently large number of input bits. This is already enough to prevent basic attacks, like those studied by Goldreich [Gol00], and we are not aware at this point of any attack.

Amplification In order to validate our assumptions even further, we present two transformations meant to enhance security of functions with block-wise locality. We consider two different techniques:

- *Amplification Technique I* produces a PRG construction with *quasi-polynomial* indistinguishability-gap (to polynomial-time distinguishers), from any *unpredictable generator* satisfying just *polynomial next-bit unpredictability* (*i.e.*, the probability of predicting any output bit given previous output bits is at most $\frac{1}{2} + \frac{1}{\mathrm{poly}(\lambda)}$, albeit for predictors in quasi-polynomial time). Though such PRGs are not strong enough for constructing IO, it suffices for constructing FE from L-linear maps; see the next section.

- *Amplification Technique II* produces a PRG construction with *sub-exponential* indistinguishability-gap, from certain special *pseudo-min-entropy-generator* whose output has sufficiently-high pseudo-min-entropy.

1.2 From Block-Wise Locality to IO and FE

We now move to an overview of our constructions from block-wise local PRGs.

IO from subexponentially secure block-wise-local PRGs Recent IO constructions from low-degree multilinear maps [LV16, Lin16b, AS16] follow a common two-step approach: They first implement appropriate FE schemes, and then transform them into an IO scheme; we refer to the second step as the (FE-to-IO) *bootstrapping step*. In more detail, they use locality-L PRGs in the bootstrapping step in order to start with FE schemes that support only computation of *degree-L polynomials*; they then show that such FE schemes can be constructed from L-linear maps. In this work, following the blueprint and technique in [Lin16b], we show how to replace the use of local PRGs with block-wise local PRGs within the bootstrapping step.

Theorem 1 (Bootstrapping using block-wise local PRGs)**.** *Let L be any positive integer. There is a construction of IO for* P/poly *from the following primitives:*

- *Public-key fully-selectively-secure (collusion-resistant) FE for degree-L polynomials whose encryption time is linear in the input length (*i.e.*, $\mathrm{poly}(\lambda)N$);*

  or with a secret-key FE scheme with the same properties, assuming additionally the subexponential hardness of LWE with subexponential modulus-to-noise ratio.

- *a PRG with block-wise locality L, block-size $\log \lambda$, and $n^{1+\alpha}$-stretch for some positive constant α.*

where both FE and PRG need to have subexponential security.

The type of secret-key FE schemes for degree-L polynomials needed above was constructed by Lin [Lin16b] assuming the SXDH assumption on L-linear maps.

Theorem 2 ([Lin16b])**.** *Let L be any positive integer. Assuming the SXDH assumption on asymmetric L-linear maps, there is a construction of secret-key fully-selectively-secure (collusion-resistant) FE schemes for degree-L polynomials whose encryption time is linear in the input length* (i.e., $\mathrm{poly}(\lambda)N$)*. Moreover, the security reduction has a polynomial security loss.*

Therefore, combining our new bootstrapping theorem with Lin's FE construction, we obtain IO from the subexponential SXDH assumption on L-linear maps, subexponentially-secure PRG with block-wise locality L, and subexponential LWE.

In the special case of $L = 2$, *public-key* FE schemes for quadratic polynomials needed above can be constructed from asymmetric bilinear maps with the Matrix Diffie-Hellman assumption and the 3-party Decision Diffie Hellman Assumption [Gay16, BCFG17]. Thus, if there is a block-wise locality 2 PRG, there is an indistinguishability obfuscator from bilinear maps (without LWE).

The power of super-polynomially secure block-wise local PRGs While constructing full-fledged IO for all polynomial-sized programs requires block-wise local PRGs with *subexponentially-security*, we ask what can be built from PRGs with weaker (slightly) *superpolynomial-security*. In particular, such PRGs can be obtained using the aforementioned *amplification technique I*, from unpredictable generator satisfying just polynomial next-bit unpredictability. As we will show, a lot can be achieved already. To this end, we first give a parameterized version of Theorem 1 that shows a tight relation between the level of security of the PRG and L-linear maps, and the class of circuits that the IO construction can obfuscate. More specifically, if the PRG and L-linear maps are $(2^{-\mathrm{i}\ell}\,\mathrm{negl})$-secure, then we can build IO schemes for circuits with $\mathrm{i}\ell$-bit inputs.

Theorem 3 (Parameterized version of Theorem 1)**.** *Let L be any positive integer. Then, there is a construction of IO for $\mathsf{P/poly}^{\mathrm{i}\ell}$ — the class of polynomial-sized circuits with $\mathrm{i}\ell$-bit inputs — from the same primitives as in Theorem 1, and if FE and PRG are $(2^{-(\mathrm{i}\ell+\kappa)}\,\mathrm{negl})$-secure, the resulting IO scheme is $(2^{-\kappa}\,\mathrm{negl})$-secure.*

Therefore, as discussed above, from slightly superpolynomially secure L-linear maps and a PRG with block-wise locality L (and subexponential LWE when $L > 2$), we obtain IO for circuits with super-logarithmic, $\omega(\log\lambda)$, length inputs, and if the primitives are quasi-polynomially secure, we obtain IO for circuits with poly-logarithmic $\log^{1+\varepsilon}(\lambda)$ length inputs. We observe that such IO schemes are sufficient for instantiating two types of natural applications of IO:

- *Type 1:* Applications where IO is used to obfuscate a circuit with short inputs. For instance, for building FHE without relying on circular security [CLTV15], and constructing succinct randomized encoding for bounded space Turning machines [BGL$^+$15]. In these applications, IO is used to obfuscate a circuit that receive as input an *index* from an arbitrary polynomial range.

- *Type 2:* Applications where the input length of the obfuscated circuit is determined by the security parameter of some other primitive. Then, by assuming exponential security of the other primitive, the input length can be made poly-logarithmic. For instance, as observed in [BNPW16, KS17], in the construction of public key encryption from one-way functions via IO, if assuming exponentially secure one-way functions, then IO for circuits with $\omega(\log\lambda)$ bit inputs suffices for the application.

Going beyond, we show that IO for circuits with super-logarithmic length inputs actually implies full-fledged functional encryption.

Theorem 4 (Functional Encryption from $\omega(\log\lambda)$-Input IO)**.** *Let* $\mathrm{i}\ell$ *be any super-logarithmic polynomial, that is,* $\mathrm{i}\ell = \omega(\log\lambda)$*. Assume IO for the class of polynomial-sized circuits with* $\mathrm{i}\ell$*-bit inputs and public key encryption, both with* $(2^{-\mathrm{i}\ell}\,\mathrm{negl})$*-security. Then, there exist collusion resistant (compact) public-key functional encryption for* P/poly*, satisfying adaptive-security.*

Combining the above two theorems, we immediately have that the existence of a PRG with block-wise locality L and L-linear maps, both with slighly super-polynomial security (and assuming subexponential LWE when $L > 2$), implies the existence of full-fledged functional encryption, and all its applications, including, for instance, non-interactive key exchange (NIKE) for unbounded users [GPSZ16], trapdoor permutations [GPSZ16], PPAD hardness [BPR15, GPS16], publicly-verifiable delegation schemes in the CRS model [PRV12], and secure traitor tracing scheme [GGH+13b, BSW06, CFN94], which further implies hardness results in differential privacy [DNR+09, Ull13].

Outline of this Paper

We review standard security notions (including security definitions for functional encryption) in Section 2. Section 3 discusses candidate constructions of block-wise local PRGs. Section 4 discusses our bootstrapping method using block-wise local PRGs. Finally, in Section 5, we discuss constructions of functional-encryption schemes in Section 5.

2 Preliminaries

Let $\mathbb{Z}$ and $\mathbb{N}$ denote the set of integers, and positive integers, respectively. Let $[n]$ denote the set $\{1, 2, \ldots, n\}$. We use $\mathcal{R}$ to denote either a ring, or an ensemble of rings $\mathcal{R} = \{\mathcal{R}_\lambda\}$, which will be clear in the context.

We denote by PPT probabilistic polynomial time Turing machines. The term *negligible* is used for denoting functions that are (asymptotically) smaller than any inverse polynomial. More precisely, a function $\nu(\star)$ from non-negative integers to reals is called *negligible* if for every constant $c > 0$ and all sufficiently large n, it holds that $\nu(n) < n^{-c}$.

We use boldface to denote vectors, for example, $\mathbf{u}, \mathbf{v}, \mathbf{c}$ etc., and use u_i, v_i, c_i to denote the i^{th} elements in the vectors.

2.1 μ-Hardness and μ-Indistinguishability

Definition 1 (μ-Hard One-Way Functions)**.** *Let* $\mu : \mathbb{N} \to [0,1]$ *be a function. A one-way function* f *is* μ*-hard if for every family of polynomial-sized adversaries* $\{A_\lambda\}_{\lambda\in\mathbb{N}}$*, and every sufficiently large security parameter* $\lambda \in \mathbb{N}$*, it holds that*

$$\Pr[x \xleftarrow{\$} \{0,1\}^n;\ y = f(x)\ :\ f(A_\lambda(y)) = y] \le \mu(\lambda)$$

Definition 2 (μ-indistinguishability)**.** *Let* $\mu : \mathbb{N} \to [0,1]$ *be a function. A pair of distribution ensembles* $\{X_\lambda\}_{\lambda\in\mathbb{N}}$*,* $\{Y_\lambda\}_{\lambda\in\mathbb{N}}$ *are* μ*-indistinguishable if for every family of polynomial-sized distinguishers* $\{D_\lambda\}_{\lambda\in\mathbb{N}}$*, and every sufficiently large security parameter* $\lambda \in \mathbb{N}$*, it holds that*

$$|\Pr[x \xleftarrow{\$} X_\lambda : D(1^\lambda, x, z) = 1] - \Pr[y \xleftarrow{\$} Y_\lambda : D(1^\lambda, y, z) = 1]| \le \mu(\lambda)$$

Definition 3 (Computational and Sub-exponential Indistinguishability). *A pair of distribution ensembles $\{X_\lambda\}_{\lambda\in\mathbb{N}}$, $\{Y_\lambda\}_{\lambda\in\mathbb{N}}$ are computationally indistinguishable if they are $1/p$-indistinguishable for every polynomial p, and are sub-exponentially indistinguishable if they are μ-indistinguishable for some sub-exponentially small $\mu(\lambda) = 2^{\lambda^\varepsilon}$ with a constant $\varepsilon > 0$.*

Note that the above definition of sub-exponential indistinguishability is weaker than standard sub-exponential hardness assumptions that consider distinguishers running in sub-exponential *time*.

Below, we provide definitions of standard cryptographic primitives using the terminology of μ-indistinguishability, which implicitly defines variants with polynomial or sub-exponential security. As a matter of convention, we will drop μ when μ is a negligible function, and say sub-exponential security when μ is a sub-exponentially small function.

2.2 Indistinguishability Obfuscation

We recall the notion of indistinguishability obfuscation for a class of circuit defined by [BGI+01b].

Definition 4 (Indistinguishability Obfuscator ($i\mathcal{O}$) for a circuit class). *A uniform* PPT *machine $i\mathcal{O}$ is an indistinguishability obfuscator for a class of circuits $\{\mathcal{C}_\lambda\}_{\lambda\in\mathbb{N}}$, if the following conditions are satisfied:*

Correctness: *For all security parameters $\lambda \in \mathbb{N}$, for every $C \in \mathcal{C}_\lambda$, and every input x, we have that*

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

where the probability is taken over the coin-tosses of the obfuscator $i\mathcal{O}$.

μ-Indistinguishability: *For every ensemble of pairs of circuits $\{C_{0,\lambda}, C_{1,\lambda}\}_{\lambda\in\mathbb{N}}$ satisfying that $C_{b,\lambda} \in \mathcal{C}_\lambda$, $|C_{0,\lambda}| = |C_{1,\lambda}|$, and $C_{0,\lambda}(x) = C_{1,\lambda}(x)$ for every x, the following ensembles of distributions are μ-indistinguishable:*

$$\left\{C_{1,\lambda}, C_{2,\lambda}, i\mathcal{O}(1^\lambda, C_{1,\lambda})\right\}_{\lambda\in\mathbb{N}}$$
$$\left\{C_{1,\lambda}, C_{2,\lambda}, i\mathcal{O}(1^\lambda, C_{2,\lambda})\right\}_{\lambda\in\mathbb{N}}$$

Definition 5 (IO for P/poly). *A uniform* PPT *machine $i\mathcal{O}_{\mathsf{P}/\mathsf{poly}}(\star,\star)$ is an indistinguishability obfuscator for* P/poly *if it is an indistinguishability obfuscator for the class $\{\mathcal{C}_\lambda\}_{\lambda\in\mathbb{N}}$ of circuits of size at most λ.*

2.2.1 $\mathrm{i}\ell$-bit-Input IO

In this work, we consider IO for polynomial-sized circuits with bounded input-length. Formally, we define IO for $\mathrm{i}\ell$-bit-input circuits, or referred to as $\mathrm{i}\ell$-bit-input IO, as follows.

Definition 6 ($\mathrm{i}\ell$-bit-input IO for P/poly). *A uniform* PPT *machine $i\mathcal{O}(\star,\star)$ is an $\mathrm{i}\ell$-bit-input indistinguishability obfuscator for* P/poly *if it is an indistinguishability obfuscator for the class $\left\{\mathcal{C}_\lambda^{\mathrm{i}\ell}\right\}_{\lambda\in\mathbb{N}}$ of circuits with size at most λ and input-length at most $\mathrm{i}\ell(\lambda) \le \lambda$.*

2.3 Puncturable Pseudo-Random Functions

We recall the definition of puncturable pseudo-random functions (PPRF) from [SW14]. Since in this work, we only use puncturing at one point, the definition below is restricted to puncturing only at one point instead of at a polynomially many points.

Definition 7 (Puncturable PRFs)**.** *Let n be a computable polynomial. A puncturable family of PRFs with domains $\{0,1\}^{n(\lambda)}$ is given by a triple of uniform PPT machines* $\mathsf{PPRF} = (\mathsf{PRF.Gen}, \mathsf{PRF.Punc}, \mathsf{F})$ *satisfying the following conditions:*

Correctness: *For every $\lambda \in \mathbb{N}$, and every output K of* $\mathsf{PRF.Gen}(1^\lambda)$*, every input $i \in \{0,1\}^{n(\lambda)}$, and $K\{i\} = \mathsf{PRF.Punc}(K, i)$, we have that $\mathsf{F}(K\{i\}, x) = \mathsf{F}(K, x)$ for all $x \neq i$.*

μ-pseudorandomness at punctured point: *For every ensemble $\left\{i_\lambda \in \{0,1\}^{n(\lambda)}\right\}$, the following ensembles (where $i = i_\lambda$) are μ-indistinguishable.*

$$\left\{K \xleftarrow{\$} \mathsf{PRF.Gen}(1^\lambda), K\{i\} = \mathsf{PRF.Punc}(K, i) \;:\; K\{i\}, i, \mathsf{F}(K, i)\right\}$$
$$\left\{K \xleftarrow{\$} \mathsf{PRF.Gen}(1^\lambda), K\{i\} = \mathsf{PRF.Punc}(K, i) \;:\; K\{i\}, i, U_\lambda\right\}$$

As observed by [BW13, BGI14, KPTZ13], the GGM tree-based construction of PRFs [GGM86] from one-way functions yields PPRFs. Furthermore, their construction incurs only a polynomial security loss, and hence, if the underlying one-way functions are μ-hard, then the resulting PPRF is μ-pseudorandom.

2.4 Randomized Encodings

In this section, we recall the traditional definition of randomized encodings with simulation security [IK02, AIK06].

Definition 8 (Randomized encoding scheme for circuits)**.** *A randomized encoding scheme* $\mathbf{RE}$ *consists of two PPT algorithms,*

- $\hat{C}_x \xleftarrow{\$} \mathsf{REnc}(1^\lambda, C, x)$*: On input a security parameter 1^λ, circuit C, and input x,* REnc *generates an encoding $\hat{C}_x$.*
- $y = \mathsf{REval}(\hat{C}_x)$*: On input $\hat{C}_x$ produced by* REnc*,* REval *outputs y.*

Correctness: *The two algorithms* REnc *and* REval *satisfy the following correctness condition: For all security parameters $\lambda \in \mathbb{N}$, circuit C, input x, it holds that,*

$$\Pr[\hat{C}_x \xleftarrow{\$} \mathsf{REnc}(1^\lambda, C, x) : \ \mathsf{Eval}(\hat{C}_x) = C(x)] = 1$$

μ-Simulation Security: *There exists a PPT algorithm* RSim*, such that, for every ensemble $\{C_\lambda, x_\lambda\}_\lambda$ where $|C_\lambda|, |x_\lambda| \leq \mathrm{poly}(\lambda)$, the following ensembles are μ-indistinguishable for all $\lambda \in N$.*

$$\left\{\hat{C}_x \xleftarrow{\$} \mathsf{REnc}(1^\lambda, C, x) : \hat{C}_x\right\}_{\lambda \in \mathbb{N}}$$
$$\left\{\hat{C}_x \xleftarrow{\$} \mathsf{RSim}(1^\lambda, C(x), 1^{|C|}, 1^{|x|}) : \hat{C}_x\right\}_{\lambda \in \mathbb{N}}$$

where $C = C_\lambda$ and $x = x_\lambda$.

Furthermore, let $\mathcal{C}$ be a complexity class, we say that randomized encoding scheme $\mathbf{RE}$ *is in $\mathcal{C}$, if the encoding algorithm* REnc *can be implemented in that complexity class.*

2.5 Functional Encryption

We provide the definition of a public-key functional encryption (FE) scheme with indistinguishability-based security which originally appeared in [BSW12, O'N10]. Below we define public key FE first, and then note the difference with secret key FE.

2.5.1 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 of sets. Let $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda\in\mathbb{N}}$, where every function in the set $\mathcal{F}_\lambda$ maps inputs in $\mathcal{X}_\lambda$ to outputs in $\mathcal{Y}_\lambda$.

A public-key functional encryption scheme $\mathbf{FE}$ for $\{\mathcal{F}_\lambda\}_{\lambda\in\mathbb{N}}$ consists of four PPT algorithms $(\mathsf{FE.Setup}, \mathsf{FE.KeyGen}, \mathsf{FE.Enc}, \mathsf{FE.Dec})$.

- *Setup:* $\mathsf{FE.Setup}(1^\lambda, \mathsf{pp})$ is an algorithm that on input a security parameter and some public parameter (*e.g.*, description of bilinear pairing groups) outputs a master public key and a master secret key $(\mathsf{MPK}, \mathsf{MSK})$.
- *Key Generation:* $\mathsf{FE.KeyGen}(\mathsf{MSK}, f)$ on input the master secret key MSK and the description of a function $f \in \mathcal{F}_\lambda$, outputs a secret key SK_f.
- *Encryption:* $\mathsf{FE.Enc}(\mathsf{MPK}, x)$ on input the master public key MPK and a message $x \in \mathcal{X}_\lambda$, outputs an encryption CT of x.
- *Decryption:* $\mathsf{FE.Dec}(\mathsf{SK}, \mathsf{CT})$ on input the secret key associated with f and an encryption of x, outputs $y \in \mathcal{Y}_\lambda$.

Correctness: We define perfect correctness here. For every λ, $f \in \mathcal{F}_\lambda$, $x \in \mathcal{X}_\lambda$, it holds that,

$$\Pr\left[\begin{array}{c}(\mathsf{MPK}, \mathsf{MSK}) \xleftarrow{\$} \mathsf{FE.Setup}(1^\lambda, \mathsf{pp}) \\ \mathsf{CT} \xleftarrow{\$} \mathsf{FE.Enc}(\mathsf{MPK}, x) \\ \mathsf{SK} \xleftarrow{\$} \mathsf{FE.KeyGen}(\mathsf{MSK}, f)\end{array} : f(x) = \mathsf{FE.Dec}(\mathsf{SK}, \mathsf{CT})\right] = 1$$

Indistinguishability Security. Indistinguishability security of a functional encryption requires that no adversary can distinguish the FE encryption of one input x_0 from that of another x_1, if the adversary only obtains secret keys for functions that yield the same outputs on x_0 and x_1, that is, for every secret key SK_f, it holds that $f(x_0) = f(x_1)$. In the adaptive setting, the two challenge inputs (x_0, x_1) and all functions f are chosen adaptively by the adversary. In the weaker fully-selective setting, the adversary is restricted to choose (x_0, x_1) and all functions f statically.

Definition 9 (Adap-security)**.** *A public-key FE scheme* $\mathbf{FE} = (\mathsf{FE.Setup}, \mathsf{FE.KeyGen}, \mathsf{FE.Enc}, \mathsf{FE.Dec})$ *for* $\{\mathcal{F}_\lambda\}_{\lambda\in\mathbb{N}}$ *is* μ-Adap-secure*, if for every* PPT *adversary* A*, and every sufficiently large security parameter* $\lambda \in \mathbb{N}$*, the adversary's advantage in the following games is bounded by* $\mu(\lambda)$

$$\mathsf{Advt}_A^{\mathbf{FE}} = \left|\Pr[\mathsf{Adap}_A^{\mathbf{FE}}(1^\lambda, 0) = 1] - \Pr[\mathsf{Adap}_A^{\mathbf{FE}}(1^\lambda, 1) = 1]\right| \leq \mu(\lambda)$$

$\mathsf{Adap}_A^{\mathbf{FE}}(1^\lambda, b)$ *proceeds as follows:*

1. **Key Generation.** *The challenger* CH *samples* $(\mathsf{MPK}, \mathsf{MSK}) \xleftarrow{\$} \mathsf{FE.Setup}(1^\lambda, \mathsf{pp})$ *and sends* MPK *to the adversary.*

2. **Function Queries I.** *Repeat the following for an arbitrary number of times determined by A: Upon A choosing a function query $f \in \mathcal{F}_\lambda$, CH sends A a function key* $\mathsf{SK}_f \xleftarrow{\$} \mathsf{FE.KeyGen}(\mathsf{MSK}, f)$.

3. **Message Queries.** *Upon A choosing a pair of messages (x_0, x_1), CH sends A a ciphertext* $\mathsf{CT} \xleftarrow{\$} \mathsf{FE.Enc}(\mathsf{MPK}, x_b)$.

4. **Function Queries II.** *Repeat the second step, for an arbitrary number of times determined by A.*

5. *Finally A outputs a bit b' which is also the output of the experiment.*

Restriction: *Every function query f must satisfy that $f(x_0) = f(x_1)$.*

Definition 10 (Full-Sel-security)**.** *We say that* $\mathbf{FE}$ *is* μ-Full-Sel-*secure if the condition in Definition 9 holds for modified experiments* $\mathsf{Full\text{-}Sel}_A^{\mathbf{FE}}(1^\lambda, b)$ *that proceeds identically to* $\mathsf{Adap}_A^{\mathbf{FE}}(1^\lambda, b)$ *except that the adversaries choose challenge messages (x_0, x_1) and all function queries $\{f\}$ at the beginning of the experiment.*

Note that our notion of fully-selective security is weaker than the notion of selective security in some papers in the literature (*e.g.*, [GKP+13, ABSV15]), which only requires the adversaries to choose challenge inputs x_0, x_1 statically, but allows the adversaries to choose challenge function inputs adaptively. Intuitively, the notion of fully-selective security is sufficient for applications that are non-interactive, for instance, building IO from FE as in [BV15, AJ15].

Definition 11 (1-key FE)**.** *We say that* $\mathbf{FE}$ *is a* μ-Adap-*secure (or* μ-Full-Sel-*secure) 1-key FE scheme if it satisfies the security requirements in Definition 9 (or, respectively, Definition 10) against adversaries that ask for at most one function key query.*

2.5.2 FE for P/poly, NC^1 and Compactness

Definition 12 (FE schemes for families of function classes)**.** *Let $\mathbb{F} = \{\mathcal{F}^I\}_{I \in \mathcal{I}}$ be a family of function classes. We say that $\mathcal{FE} = \{\mathbf{FE}^I\}_{I \in \mathcal{I}}$ is a family of (1-key) FE schemes for $\mathbb{F}$ with* μ-Adap-*security or* μ-Full-Sel-*security if for every function class $\mathcal{F}^I = \{\mathcal{F}^I_\lambda\}_{\lambda \in \mathbb{N}}$, $\mathbf{FE}^I$ is a (1-key) FE scheme for $\mathcal{F}^I$ with* μ-Adap-*security or* μ-Full-Sel-*security.*

Moreover, define the following special cases:

- **FE for** P/poly *is a family of FE schemes for $\mathbb{F} = \{\mathcal{F}^{N,D,S}\}_{N \in \mathcal{N}, D \in \mathcal{D}, S \in \mathcal{S}}$, where $\mathcal{N}, \mathcal{D}, \mathcal{S}$ are the sets of all polynomials and $F^{N,D,S}$ is the class of binary functions that can be computed by circuits with $N(\lambda)$-bit inputs, $S(\lambda)$ size, and $D(\lambda)$ depth.*

- **FE for** NC^1 *is a family of FE schemes for $\mathbb{F} = \{\mathcal{F}^{N,D,S}\}_{N \in \mathcal{N}, D \in \mathcal{D}, S \in \mathcal{S}}$ as defined above but with $\mathcal{D}$ the set of all logarithmic functions.*

Compactness In the above definition of families of FE schemes, algorithms in scheme $\mathbf{FE}^{N,D,S}$ could run in polynomial time depending on polynomials N, D, S. In the literature, stronger efficiency requirements have been considered. In particular, the works of [AJ15, BV15] defined compact FE schemes for NC^1, which requires the encryption time to be independent of the circuit size S of the functions.

Definition 13 (Compactness of FE schemes for NC^1)**.** *Let $\mathcal{FE} = \{\mathbf{FE}^{N,D,S}\}$ be a family of FE schemes for* NC^1.

Compactness: *We say that the functional encryption scheme $\mathcal{FE}$ is compact if for every logarithmic function D, there is a polynomial p, such that, for every polynomials N, S, the encryption algorithm of $\mathbf{FE}^{N,D,S}$ runs in time $p(\lambda, N(\lambda), \log S(\lambda))$.*

$(1-\varepsilon)$**-Sublinear Compactness (a.k.a.** $(1-\varepsilon)$**-Weakly Compactness):** *We say that $\mathcal{FE}$ is $(1-\varepsilon)$-sublinearly compact, if for every logarithmic function D, there is a polynomial p, such that, for every polynomials N, S, the encryption algorithm of $\mathbf{FE}^{N,D,S}$ runs in time $p(\lambda, N(\lambda)) \cdot S(\lambda)^{1-\varepsilon}$.*

2.6 Zero-Testing FE for Arithmetic Functions

For any ring $\mathcal{R}$, we refer to functions mapping from $\mathcal{R}^*$ to $\mathcal{R}^*$ as arithmetic functions in $\mathcal{R}$. Many previous works (*e.g.* [ABCP15, BJK15]) constructed FE schemes for classes of arithmetic functions in $\mathcal{R}$ with a relaxed correctness guarantee, namely, decryption does not reveal the output (in $\mathcal{R}$) entirely, but only reveals whether the output is zero or not. We refer to this relaxed correctness guarantee as *zero-testing correctness*, and FE schemes with such relaxed correctness as *zero-testing FE*. We stress that though the correctness requirement is relaxed, the security requirements, namely IND-security and function hiding, remain the same. Therefore, zero-testing FE is strictly weaker than standard FE.

Definition 14 (Zero-testing FE)**.** *Let $\mathcal{R} = \{\mathcal{R}_\lambda\}$ be an ensemble of rings, and $\{\mathcal{F}_\lambda\}$ a class of functions where $\mathcal{F}_\lambda$ maps from $\mathcal{X}_\lambda \subseteq \mathcal{R}_\lambda^*$ to $\mathcal{Y}_\lambda \subseteq \mathcal{R}_\lambda^*$. We say that $\mathbf{FE}$ is a (1-key) zero-testing FE scheme for $\{\mathcal{F}_\lambda\}$ with μ-*Adap*-security or μ-*Full-Sel*-security, if it is a FE scheme for $\{\mathcal{F}_\lambda\}$ with the same security guarantee as in Definition 9 or 10 respectively, and the following relaxed correctness guarantee.*

- **Zero-Testing Correctness:** *For every λ, $f \in \mathcal{F}_\lambda$, $x \in \mathcal{X}_\lambda$, it holds that,*

$$\Pr\left[\begin{array}{c}(\mathsf{MPK},\mathsf{MSK}) \xleftarrow{\$} \mathsf{FE.Setup}(1^\lambda,\mathsf{pp})\\ \mathsf{CT} \xleftarrow{\$} \mathsf{FE.Enc}(\mathsf{MPK},x)\\ \mathsf{SK} \xleftarrow{\$} \mathsf{FE.KeyGen}(\mathsf{MSK},f)\end{array} : \mathsf{ZT}(f(x)) = \mathsf{FE.Dec}(\mathsf{SK},\mathsf{CT})\right] = 1$$

where ZT is a predicate that outputs 1 iff its input is the zero element in $\mathcal{R}_\lambda$, and in the case of secret key FE, $\mathsf{MPK} = \mathsf{MSK}$.

Zero-Testing FE for Degree-d Polynomials

Definition 15 (Zero-testing FE schemes for families of arithmetic function classes)**.** *Let $\mathbb{F} = \{\mathcal{F}^I\}_{I \in \mathcal{I}}$ be a family of arithmetic function classes. A family $\mathcal{FE} = \{\mathbf{FE}^I\}_{I\in\mathcal{I}}$ of (1-key) zero-testing FE schemes for $\mathbb{F}$ is defined identically as in Definition 12 except that every scheme $\mathbf{FE}^I$ has zero-testing correctness.*

Moreover, define the following special cases:

- **Zero-testing FE for degree-d polynomials in $\mathcal{R}$** *is a family of zero-testing FE schemes for $\mathbb{F} = \{\mathcal{F}^N\}$ where where $\mathcal{F}^N$ is the set of degree-d polynomials mapping from $\mathcal{R}_\lambda^{N(\lambda)}$ to $\mathcal{R}_\lambda$.*

Definition 16 (Linear efficiency)**.** *Let $\mathcal{FE} = \{\mathbf{FE}^N\}$ be a family of FE schemes for degree-d polynomials or inner products in $\mathcal{R}$. We say that $\mathcal{FE}$ has* **linear efficiency** *if there exists a polynomial function p, such that, for every polynomial N, the encryption algorithm of $\mathbf{FE}^N$ runs in time $N(\lambda)\,\mathrm{poly}(\lambda)$.*

In the rest of the paper, whenever we talk about FE for arithmetic functions, in particular, FEs for degree-d polynomials, over a family of non-binary ring $\mathcal{R}$, we mean by default a zero-testing FE.

2.7 Degree-D Asymmetric Multilinear Maps with SXDH Assumption

Introduced by Boneh and Silverberg [BS02], asymmetric Multilinear Maps (MMaps) naturally generalize asymmetric bilinear maps to higher degree. Let $\mathcal{G}$ denote a group generator that on input 1^λ outputs $(p, G_1, \cdots, G_D,$
$G_{D+1}, \mathbf{pair})$, where $G_1, \cdots, G_D, G_{D+1}$ are cyclic groups with order p (prime or composite). G_1 to G_D are referred to as the source groups and G_{D+1} the target group. Assume without loss of generality that the description of the source groups contain generators $g_1, \cdots, g_D$ of $G_1, \cdots, G_D$. In addition, the following properties hold.

- *Admissible:* $\mathbf{pair} : G_1 \times \cdots \times G_D \to G_{D+1}$ is efficiently computable and $g_{D+1} = \mathbf{pair}(g_1 \cdots, g_D)$ generates G_{D+1}.
- *Multilinear:* For any $a_1, \cdots, a_D \in \mathbb{Z}_p$, $\mathbf{pair}(g_1^{a_1}, \cdots, g_D^{a_D}) = \mathbf{pair}(g_1, \cdots, g_D)^{a_1 a_2 \cdots a_D} = g_{D+1}^{a_1 a_2 \cdots a_D}$.

We denote by $\mathcal{R}_\lambda = (\mathbb{Z}_p, +, \times)$ the ring corresponding to the exponent space of these multilinear pairing groups.

The Bracket Notation For clarity of notions, we use the following bracket notations to denote group elements.

$$\forall l \in [D+1], \quad [a]_l = g_l^a$$

We refer to $[a]_l$ as an encoding of a in group G_l, or with label l. Under this notation, the generator in group $l \in [D+1]$ is represented as $[1]_l = g_l$. We also use the following vector notation to represent vectors of group elements succinctly: For any $\mathbf{v} = (v_1, \cdots, v_m) \in \mathbb{Z}_p^m$, and $l \in \{0, 1, T\}$:

$$[\mathbf{v}]_l = [v_1]_l \cdots [v_m]_l$$

The SXDH Assumption The SXDH assumption states that the standard DDH assumption holds in each of the source groups. Formally, for every source group G_l for $l \in [D]$, the following two ensembles are μ-indistinguishable.

$$\left\{ \mathsf{pp} = (p, G_1, \cdots G_D, G_{D+1}, \mathbf{pair}) \xleftarrow{\$} \mathcal{G}(1^\lambda),\ a, b \xleftarrow{\$} \mathbb{Z}_p \ :\ (\ \mathsf{pp},\ [a]_l, [b]_l, [ab]_l\ ) \right\}_\lambda$$
$$\left\{ \mathsf{pp} = (p, G_1, \cdots, G_D, G_{D+1}, \mathbf{pair}) \xleftarrow{\$} \mathcal{G}(1^\lambda),\ a, b, r \xleftarrow{\$} \mathbb{Z}_p \ :\ (\ \mathsf{pp},\ [a]_l, [b]_l, [r]_l\ ) \right\}_\lambda$$

3 Block-Wise Local PRGs

In this section, we introduce the notion of a block-wise local PRG. We start with formal definitions, in Section 3.1, which we refer to throughout the rest of the paper. Then, the remaining sub-sections will discuss a graph-based framework for block-wise local functions, and discuss candidates.

3.1 Pseudorandom Generators, Locality, and Block-Wise Locality

Definition 17 (Family of Pseudo-Random Generators (PRGs)). *Let n and m be polynomials. A family of $(n(\lambda), m(\lambda))$-PRG is an ensemble of distributions $\mathbf{PRG} = \{\mathbf{PRG}_\lambda\}$ satisfying the following properties:*

Syntax: *For every $\lambda \in \mathbb{N}$, every* PRG *in the support of $\mathbf{PRG}_\lambda$ defines a function mapping $n(\lambda)$ bits to $m(\lambda)$ bits.*

Efficiency: *There is a uniform Turning machine M satisfying that for every $\lambda \in \mathbb{N}$, every* PRG *in the support of* $\mathbf{PRG}_\lambda$*, and every* $x \in \{0,1\}^{n(\lambda)}$, $M(\mathrm{PRG}, x) = \mathrm{PRG}(x)$.

μ-Indistinguishability: *The following ensembles are μ-indistinguishable*

$$\left\{\mathrm{PRG} \xleftarrow{\$} \mathbf{PRG}_\lambda;\ s \xleftarrow{\$} \{0,1\}^{n(\lambda)} : (\mathrm{PRG}, \mathrm{PRG}(s)\right\}_{\lambda\in\mathbb{N}}$$
$$\approx_\mu \left\{\mathrm{PRG} \xleftarrow{\$} \mathbf{PRG}_\lambda;\ r \xleftarrow{\$} \{0,1\}^{m(\lambda)} : (\mathrm{PRG}, r)\right\}_{\lambda\in\mathbb{N}}$$

Definition 18 (Locality of PRGs). *Let n, m, and L be polynomials. We say that a family of PRGs* $\mathbf{PRG}$ *has locality L if for every $\lambda \in \mathbb{N}$ and every* PRG *in the support of* $\mathbf{PRG}_\lambda$*, every output bit of* PRG *depends on at most $L(\lambda)$ input bits.*

Definition 19 (Block-Wise Locality of PRGs). *Let n, m, L, and ℓ be polynomials. We say that a family of $(n(\lambda)\ell(\lambda), m(\lambda))$-PRGs has block-wise locality-$(L(\lambda), \ell(\lambda))$ if for every λ and every* PRG *in the support of* $\mathbf{PRG}_\lambda$*, inputs of* PRG *are viewed as $n(\lambda) \times \ell(\lambda)$ matrices of bits, and every output bit of* PRG *depends on input bits contained in at most $L(\lambda)$ columns.*

In this work, we consider PRGs with constant locality and constant block-wise locality; we generally refer to them as local PRGs or block-wise local PRGs respectively.

3.2 Graph-Based Block-Wise local Functions

In this section, we discuss candidate PRGs with block-wise locality d, where d can be as small as two. Here, we start with the notational framework and then move on to discussing concrete assumptions on them in Section 3.3.

Goldreich's function We will consider local functions based on Goldreich's construction [Gol00], which have been the subject for extensive study (cf. e.g. Applebaum's survey [App15]).

Recall first that an $[n, m, d]$-hypergraph is a collection $G = (S_1, \ldots, S_m)$ where the *hyerpedges* S_i are elements of $[n]^d$, i.e., $S_i = (i_1, \ldots, i_d)$, where $i_j \in [n]$ (note that we allow for potential repetitions, merely for notational convenience). We use hypergraphs to build functions as follows.

Definition 20 (Goldreich's function). *Let $\mathbf{G} = \{\mathbf{G}_\lambda\}_{\lambda\in\mathbb{N}}$ be an ensemble such that $\mathbf{G}_\lambda$ is a distribution on $[n(\lambda), m(\lambda), d(\lambda)]$-hypergraphs, for polynomial functions m, n, d. Also let $P = \{P_\lambda\}_{\lambda\in\mathbb{N}}q$ be a family of predicates, where P_λ operates on $d(\lambda)$-bit strings. Then, define the function ensemble $\mathbf{GF}^{\mathcal{G},P} = \{\mathbf{GF}^{\mathcal{G},P}_\lambda\}_{\lambda\in\mathbb{N}}$, where $\mathbf{GF}^{\mathcal{G},P}_\lambda$ samples first a graph $G = (S_1, \ldots, S_m) \xleftarrow{\$} \mathbf{G}_\lambda$, and then outputs the function $\mathrm{GF}_{G,P} : \{0,1\}^n \to \{0,1\}^m$ such that for all n-bit x,*

$$\mathrm{GF}_{G,P}(x) = (y_1, \ldots, y_m)\ , \quad y_i = P(x[S_i])\ ,$$

where $x[S]$ denotes the d-bit sub-string obtained by concatenating the bits at positions indexed by S.[1]

[1]The notion could be block-wise to the cases where predicates are drawn by a distribution, and possibly differ from each output bit. We are going to dispense with such extensions, which are straightforward but easily lead to notational overhead.

Functions with block-wise locality We want to extend the notation used above to consider the case where an edge of G does not solely give a pointer to individual bits to be injected in the computation, but rather, to "chunks" consisting of ℓ-bit strings, and the predicate is applied to the concatenation of these bits. The resulting function clearly then satisfies block-wise locality d with block size ℓ.

Definition 21 (Block-wise local graph-based function). *Let $\mathbf{G} = \{\mathbf{G}_\lambda\}_{\lambda \in \mathbb{N}}$ be such that $\mathbf{G}_\lambda$ is a distribution on $[n(\lambda), m(\lambda), d(\lambda)]$-hypergraphs, for polynomial functions m, n, d. Also let $\ell(\lambda)$ be a polynomial function, and $P = \{P_\lambda\}_{\lambda \in \mathbb{N}}$ a family of predicates, where P_λ operates on $(d(\lambda) \times \ell(\lambda))$-bit strings. Then, define the function ensamble $\mathbf{GF}^{\mathcal{G},P,\ell} = \{\mathbf{GF}_\lambda^{\mathcal{G},P,\ell}\}_{\lambda \in \mathbb{N}}$, where $\mathbf{GF}_\lambda^{\mathcal{G},P,\ell}$ samples first a graph $G = (S_1, \ldots, S_m) \overset{\$}{\leftarrow} \mathbf{G}_\lambda$, and then outputs the function $\mathrm{GF}_{G,P,\ell} : \{0,1\}^{n \cdot \ell} \to \{0,1\}^m$ such that for all $(n \times \ell)$-bit inputs $\mathbf{x} = (\mathbf{x}[1], \ldots, \mathbf{x}[n])$, where $\mathbf{x}[1], \ldots, \mathbf{x}[n] \in \{0,1\}^\ell$,*

$$\mathrm{GF}_{G,P,\ell}(x) = (y_1, \ldots, y_m) \;, \quad y_i = P(\mathbf{x}[S_i]) \;,$$

where $\mathbf{x}[S]$ denotes the $d \cdot \ell$-bit sub-string obtained by concatenating ℓ-bit input chunks indexed by S.

We typically refer to the graph G describing $\mathrm{GF}_{G,P,\ell}$ as the *base graph*. This is because $\mathrm{GF}_{G,P,\ell}$ can be seen as a special case of Goldreich's function defined above, for a suitable graph. Namely, the base graph G can be extended to an $[n \cdot \ell, m, d\ell]$-hypergraph $\overline{G}$ naturally, where each edge $S_i = (i_1, \ldots, i_d)$ from G is mapped into a new hyper-edge $\overline{S}_i$ with $d \cdot \ell$ elements such that

$$\overline{S}_i = ((i_1 - 1) \cdot \ell + 1, \ldots, i_1 \cdot \ell, \cdots, (i_d - 1) \cdot \ell + 1, \ldots, i_d \cdot \ell) \;,$$

then clearly $\mathrm{GF}_{G,P,\ell} = \mathrm{GF}_{\overline{G},P,1} = \mathrm{GF}_{\overline{G},P}$. This view will be convenient to connect back to the body of work on studying the security of Goldreich's function on suitable graphs, for which our block-wise local designs serve as a special case.

Expansion properties In general, we will want to instantiate our framework with functions where the base graph G is a good expander graph. Recall the following.

Definition 22. *$G = (S_1, \ldots, S_m)$ is a (k, c)-expander (or, equivalently, is (k, c)-*expanding*) if for all sets $J \subseteq [m]$ with $|J| \leq k$, we have $\left|\bigcup_{j \in J} S_j\right| \geq c \cdot |J|$.*

Ideally, we will want in fact $\overline{G}$ to be a good expander (in order to resort to large body of analyses for such functions). This will follow by making the base graph a good expander. In particular, the following simple fact stems from the observation that when going from G to $\overline{G}$, we have $\left|\overline{S}_j\right| = \ell \left|S_j\right|$, and hence the (relative) expansion factors of G and $\overline{G}$ are identical.

Lemma 1. *Let G be an $[n, m, d]$-hypergraph which is $(k, (1 - \gamma)d)$-expanding. Then, for any block-size ℓ, the resulting $[n \cdot \ell, m, d\ell]$-hypergraph $\overline{G}$ is $(k, (1 - \gamma)d\ell)$-expanding.*

In general, if we have high degree (say $O(\log \lambda)$), we can prove the existence (at least probabilistically) of very good expanders with expansion rate very close to the degree. Unfortunately, our construction of $\overline{G}$ imposes some structure, and the actual expansion factor is dictated by the graph G with much lower degree d. The following lemma establishes the existence of good expander graphs, which we summarize below in a corollary with more useful parameters. While the proof of the lemma is folklore (we take notational inspiration from the one in [ABR16]), we give a more careful analysis tailored at a tight characterization for low degrees, including $d = 2$.

Lemma 2 (Strong expansion lemma). *Let $d \geq 2$, and let $\gamma \in (0,1)$ and $\beta \in (0, 1/2)$ be such that $d\gamma = 1 + \beta$. Further, let $1 \leq \Delta \leq n^{\beta}/\log(n)$. Then, there exists a constant $\alpha > 0$ such that a random $[n, m = \Delta n, d]$-hypergraph G is a $(k = \alpha n/\Delta^{1/\beta}, d(1-\gamma))$-expander with probability $1 - o(1)$.*

Proof. We pick a random $[n, \Delta n, d]$-hypergraph $G = (S_1, \ldots, S_m)$. Then, by a standard argument, with $c = (1-\gamma)d$, the probability that G is not (k, c)-expanding is upper bounded by

$$\begin{aligned}
\sum_{r=1}^{k} \binom{\Delta n}{r} \binom{n}{c \cdot r} \left(\frac{cr}{n}\right)^{d \cdot r} &\leq \sum_{r=1}^{k} \left(\frac{e\Delta n}{r}\right)^{r} \left(\frac{en}{c \cdot r}\right)^{cr} \left(\frac{cr}{n}\right)^{d \cdot r} \\
&= \sum_{r=1}^{k} \left(\frac{e^{c+1}\Delta n}{r}\right)^{r} \left(\frac{cr}{n}\right)^{\gamma d r} \\
&= \sum_{r=1}^{k} \left(\frac{e^{c+1}\Delta c^{\gamma d}}{(n/r)^{\gamma d - 1}}\right)^{r} = \sum_{r=1}^{k} \left(\frac{C \cdot \Delta}{(n/r)^{\gamma d - 1}}\right)^{r} ,
\end{aligned}$$

where $C = C_{\gamma,d} = e^{c+1}c^{\gamma d}$ is a constant which only depends on γ and d. Now, with $p_r = \left(\frac{C \cdot \Delta}{(n/r)^{\gamma d - 1}}\right)$, note that because $1 \leq \Delta \leq n^{\beta}/\log(n)$ and $\gamma d - 1 = \beta$,

$$p_r \leq \left(\frac{C \Delta r^{\beta}}{n^{\beta}}\right)^{r} \leq \left(\frac{C r^{\beta}}{\log(n)}\right)^{r} .$$

To compute $\sum_{r=1}^{k} p_r$, we partition its summands into three different sets:

- For $r = 1, \ldots, \lfloor 1/\beta \rfloor$, we have that the sum of the p_r's in this range is $o(1)$, because $\lfloor 1/\beta \rfloor \geq 1$ is constant.
- For $\lfloor 1/\beta \rfloor \leq r \leq 10 \log(n)$, then we have
$$p_r \leq \left(\frac{C 10^{\beta} \log(n)^{\beta}}{\log(n)}\right)^{1/\beta} \leq O\left(\frac{1}{\log^{\frac{1}{\beta}-1}(n)}\right) ,$$
and because $1/\beta - 1 > 1$, the sum of the p_r's in this range is also $o(1)$.
- For $10 \log(n) \leq r \leq n/(2C \cdot \Delta)^{1/\beta}$, we have $p_r \leq (\frac{1}{2})^{10 \log(n)} \leq O(n^{-10})$. However, note that the number of r's in this range is at most $O(n)$, and thus the sum of the p_r's in this range is also $o(1)$.

This concludes the proof. □

Corollary 1. *For every γ and d such that $1 < \gamma d < 1.5$, and every $\eta \in (0,1)$, there exists a $[n, n^{1+\zeta}, d]$-hypergraph (for some $\zeta > 0$) which is a $(n^{1-\eta}, (1-\gamma)d)$-expander.*

Proof. With $\beta = \gamma d - 1$, set $\Delta = n^{\eta\beta}/\log(n)$. Then, $\Delta^{1/\beta} < n^{\eta}$ for large enough n, and thus $k > n^{1-\eta}$. □

3.3 Pseudorandom and Unpredictability Generators

We are interested in the question of finding $[n, m, d]$-hypergraphs for $m = n^{1+\alpha}$ and a constant $d \geq 2$ such that $\mathrm{GF}_{G,P,\ell}$ is a good PRG, for $\ell = O(\log \lambda)$. We consider a parameterized assumption on such functions (in terms of unpredictability), and discuss it briefly. Below, in Sections 3.5 and 3.6, we are then going to show how strong indistinguishability follows from (potentially) weaker versions of this assumption.

Unpredictability generator and assumptions Let $\mathbf{UG} = \{\mathbf{UG}_\lambda\}_{\lambda\in\mathbb{N}}$ be a function ensemble, where $\mathbf{UG}_\lambda$ is a distribution on functions from $n(\lambda)$ to $m(\lambda)$ bits, for some polynomial functions m and n.

Definition 23 (Unpredictability generator)**.** *We say that* $\mathbf{UG}$ *is an* (s,δ)*-unpredictability generator (or* (s,δ)*-UG, for short) if for all (non-uniform) adversaries* $A = \{A_\lambda\}_{\lambda\in\mathbb{N}}$ *with size at most* $s(\lambda)$ *and all sequences of indices* $i(\lambda) \in \{0,\ldots,i(\lambda)-1\}$*, we have*

$$\Pr\left[\begin{array}{c} x \xleftarrow{\$} \{0,1\}^{n(\lambda)} \\ \mathrm{UG} \xleftarrow{\$} \mathbf{UG}_\lambda \end{array} : A_\lambda(\mathrm{UG}, \mathrm{UG}_{\leq i(\lambda)}(x)) = \mathrm{UG}_{i(\lambda)+1}(x)\right] \leq \frac{1}{2} + \delta(\lambda) ,$$

where $\mathrm{UG}_{\leq j}(x)$ *and* $\mathrm{UG}_j(x)$ *denote the first* j *bits and the* j*-th bit of* $\mathrm{UG}(x)$*, respectively.*

Note that by a standard argument, being a (s,δ)-UG implies being a (family of) $(s, O(m\cdot\delta))$-PRGs. We now consider the following assumption, which parametrizes the fact that $\mathrm{GF}_{G,P,\ell}$ is a good PRG.

Definition 24 (BLUG-assumption)**.** *Let* $n,\ell,s : \mathbb{N}\to\mathbb{N}$*, and let* $d \geq 2$ *and* $\alpha > 0$ *be constants. Also, let* $\delta : \mathbb{N} \to [0,1]$*. Then, the* (d,ℓ)*-BLUG*(n,α,s,δ) *assumption is the assumption that there exists a family* $G = \{G_\lambda\}_{\lambda\in\mathbb{N}}$ *of* $[n(\lambda), n(\lambda)^{1+\alpha}, d]$ *hypergraphs, and a family* $P = \{P_\lambda\}_{\lambda\in\mathbb{N}}$ *of predicates on* $(d(\lambda)\times\ell(\lambda))$*-bit strings such that* $\mathbf{GF}^{G,P,\ell}$ *is an* (s,δ)*-UG.*

We are being a bit informal here, in the sense that obviously we would like $\mathbf{GF}^{G,P,\ell}$ to additionally be efficiently computable in a uniform sense. Our candidates will not have this property, as we are only able to infer the existence of suitable G's probabilistically. There are two ways of thinking about the resulting ensemble: Either non-uniformly – the graph G_λ is given as advice for security parameter λ – but usually we actually show that a $1-o(1)$ fraction of the $[n, n^{1+\alpha}, d]$-hypergraphs are good choices. In that case, we replace G with $\mathbf{G}$ where $\mathbf{G}_\lambda$ chooses a random $[n(\lambda), n(\lambda)^{1+\alpha}, d(\lambda)]$-hypergraph G, which is *bad* with vanishing probability $o(1)$. This is of course not good enough, yet the problem can often be by-passed in an application-dependent way, by considering the fact that the *end scheme* using $\mathbf{GF}^{\mathbf{G},P,\ell}$ will also be insecure with probability $o(1)$. One can then consider $\omega(1)$-instances of this scheme, each using an independent instance from $\mathbf{GF}^{\mathbf{G},P,\ell}$, and then combine them with a combiner, if it exists.

Our constructions below require $(d, O(\log(\lambda)))$-BLUG$(n,\alpha,\mathrm{poly}(\lambda), 2^{-\omega(\log\lambda)})$ to be true for some $n(\lambda) = \mathrm{poly}(\lambda)$ and $\alpha > 0$. For stronger results, we are going to replace $2^{-\omega(\log\lambda)}$ with $2^{-\lambda^\epsilon}$ for some $\epsilon > 0$. Below, in Sections 3.5 and 3.6, we will discuss whether this assumption can be implied by (qualitatively) weaker properties. We will show in particular that $(d, O(\log^{1-\varepsilon}(\lambda)))$-BLUG$(n,\alpha,2^{\omega(\log\lambda)}, 1/\lambda^{\Omega(1)})$ implies $(d, O(\log(\lambda)))$-BLUG$(n,\alpha,\mathrm{poly}(\lambda),2^{-\omega(\log\lambda)})$.

Here, we briefly discuss what can be expected to start with.

The case $d \geq 3$. For the case $d\geq 3$, a good candidate to study is the case where $\ell = O(\log(\lambda))$ and $G = \{G_\lambda\}_{\lambda\in\mathbb{N}}$ is such that G_λ is an $[n(\lambda), n(\lambda)^{1+\alpha}, d]$-hypergraph which is a good $(n^{1-\gamma}, (1-\gamma)d)$-expander where $\gamma < \frac{1}{2}$, which exists (for some suitable $\alpha > 0$) by Corollary 1. The corresponding $\overline{G}_\lambda$ are then in turn also $(n^{1-\gamma}, (1-\gamma)d\ell)$-expanders by Lemma 1.

Applebaum and Raykov [AR16] recently justify the assmption that for suitable predicates, P, the function family $\mathbf{GF}^{\overline{\mathcal{G}},P}$ is one way and a PRG against adversary running in time $2^{n^{1-\gamma}}$, which cannot succeed with probability larger than $2^{-n^{1-\gamma}}$. In the same paper, they also give a decision-to-search reduction for such functions, which however applies only for degrees where we can accommodate some γ with $3\gamma < 1$. In particular, such functions withstand existing attacks, such as myopic inversion attacks [CEMT09]. Also, the degree of P can be high, e.g., $O(\log(\lambda))$, and this prevents a number of attacks exploiting weakness of the predicate [CM01, BQ12].

Also, as we show in the next section, it is possible to adopt the techniques from [ABR16] to show that we can get good ϵ-biased genertors (for a sub-exponential ϵ) with block-wise locality $(3, 2)$. This has been the main technique in validating PRG assumptions on graph-based local functions [MST03, ABR16, OW14].

The special case $d = 2$. The case $d = 2$ is particularly important, as it does allow instantiations from bilinear maps in our applications. Note that algebraic attacks are mitigated here – in contrast to the case of plain locality, i.e., $\ell = 1$, we can set $\ell = O(\log \lambda)$ and achieve sufficiently high algebraic degree of the predicate P. However, we cannot hope to build a base graph G with expansion at least $d/2$ (however, note that for appropriate $m = n^{1+\alpha}$, we can get arbitrarily close by Corollary 1, e.g., $0.499d$ expansion). Consequently, this implies that $\overline{G}$ also has expansion less than $d\ell/2 = \ell$.

This breaks so-called *unique vertex expansion*, i.e., the property that for every subset of at most k edges, there exists an element which appears in exactly one of these edges. As a consequence of this, existing results and techniques to prove security against limited classes of attacks (e.g., [CEMT09, ABR16]) fail here, with the exception of the original attack presented in [Gol00], which only requires additive expansion, i.e., $\left|\sum_{j\in J} S_j\right| - |J| = \Omega(\lambda)$ for all sufficiently large $|J|$. However, this does not mean that some attack from such classes succeed – indeed, we leave investigating attacks as an open question.

3.4 Block-Wise local Small-Bias Generators

Several works [CM01, MST03, AL16, ABR16] have focused on studying weaker properties achieved by local generators. In particular, a standard statement towards validating their security is that of showing that the meet the definition of being a *small-bias generator*.

Definition 25. *We say* $\mathrm{SB} : \{0,1\}^n \to \{0,1\}^m$ *is an* ϵ-small biased generator *if* $\max_{J\subseteq[n],J\neq\emptyset}\left|\Pr[x \xleftarrow{\$} \{0,1\}^n \;:\; \bigoplus_{j\in J} \mathrm{SB}_j(x) = 1] - \frac{1}{2}\right| \leq \epsilon$*, where* $\mathrm{SB}_j(x)$ *denotes the* j*-th bit of* $\mathrm{SB}(x)$*.*

We show that $\mathrm{GF}_{G,Q,2}$ is a good small-biased generator for a sub-exponential ϵ, where G is an $[n, m, 3]$-hypergraph, and Q is the predicate which given three 2-bit blocks $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3$ where $\mathbf{x}_i = (x_i^l, x_i^h)$, outputs

$$Q(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3) = x_1^l \oplus x_2^l \oplus x_3^l \oplus (x_1^h \wedge x_2^h) \; .$$

Another convenient way to think about $\mathrm{GF}_{G,Q,2}$ is as

$$\mathrm{GF}_{G,Q,2}((x_1^l, x_1^h), \ldots, (x_n^l, x_n^h)) = \mathrm{GF}_{G,Q^l}(x_1^l, \ldots, x_n^l) \oplus \mathrm{GF}_{G,Q^h}(x_1^h, \ldots, x_n^j) \; ,$$

where $Q^l(x_1, x_2, x_3) = x_1 \oplus x_2 \oplus x_3$ and $Q^h(x_1, x_2, x_3) = x_1 \wedge x_2$. To show that $\mathrm{GF}_{G,Q,2}$ has small bias, the main idea is fairly straightforward. Indeed, current analyses of local small-biased generators give two separate analyses for so called "light tests" and "heavy tests", where the "weight" of a test amounts to the cardinality of $|J|$. For standard locality, withstanding both at the same time forces the graph degree to be at least five, since the predicate needs to be "non-degenerate" for the construction to withstand tests (and the theorem of [ABR16] to apply), and all predicates up to $d = 4$ *are* degenerate (cf. e.g. [CM01]). This will not be a problem here, as we only target block-wise locality, and thus effectively the predicate can be non-degenerate. The proof will in fact show that for most graphs G, GF_{G,Q^l} resists light tests, whereas GF_{G,Q^h} resists heavy tests, and thus their xor resists *all* tests for most graphs. The proof easily extends to any Q^l and Q^h which resist light and heavy tests, respectively.

Lemma 3. *For all $\delta > 0$ and $\alpha < \frac{1-\delta}{4}$, for a fraction of $1 - o(1)$ of all $[n, n^{1+\alpha}, 3]$-hypergraphs G, and Q as defined above, $\mathrm{GF}_{G,Q,2}$ is an $\left(e^{-\frac{n^\delta}{4}}\right)$-biased generator.*

Proof. Our proof relies on the machinery introduced by [ABR16]. In particular, we will distinguish between light and heavy tests, depending on whether the set $|J|$ indexing the bits to be xored is large or not. Then, for a random x, denote $y = \mathrm{GF}_{G,Q,2}(x) = y^h \oplus y^l$, where $y^h = \mathrm{GF}_{G,Q^h}(x)$ and $y^l = \mathrm{GF}_{G,Q^l}(x)$. We have

$$\bigoplus_{j\in J} y_j = \left(\bigoplus_{j\in J} y_j^h\right) \oplus \left(\bigoplus_{j\in J} y_j^l\right) ,$$

and since $\bigoplus_{j\in J} y_j^h$ is independent from $\bigoplus_{j\in J} y_j^l$, with $b^v(J) = \left|\Pr\left[x \xleftarrow{\$} \{0,1\}^n \ : \ \bigoplus_{j\in J} y_j^v = 1\right] - \frac{1}{2}\right|$ for $v \in \{l, h\}$, we have

$$b(J) := \left|\Pr\left[x \xleftarrow{\$} \{0,1\}^n \ : \ \bigoplus_{j\in J} y_j = 1\right] - \frac{1}{2}\right| \leq 2b^l(J)\cdot b^h(J) \leq 2\min\{b^l(J), b^h(J)\} . \qquad (1)$$

We are going to show that for a suitable G, $b^l(J)$ is small for light tests, i.e., small J, whereas $b^h(J)$ will be shown to be small for heavy tests. Let us start discussing the former case.

Let $\Delta = n^\alpha \leq \sqrt{n}/\log(n)$. First, assume we pick $G = (S_1, \ldots, S_m)$ as a random $[n, \Delta n, 3]$ hypergraph. We say that G is k-linear if for all sets $J \subseteq [m]$ with $|J| \leq k$, the incidence vectors $\{\mathbf{v}_j\}_{j\in J}$ of the sets $\{S_j\}_{j\in J}$ (which are in particular vectors in $\mathbb{F}_2^n$) are linearly independent. Then, [ABR16] show that G is k-linear with probability $1 - o(1)$, for $k = k(n, \Delta) = \Omega(n/\Delta^2)$. Then, [ABR16] also show in particular that in this case, because the predicate $Q^l(x_1, x_2, x_3) = x_1 \oplus x_2 \oplus x_3$ is 2-resilient (i.e., its Fourier coefficients with weight ≤ 2 are all 0) the bits $y_1^l, \ldots, y_m^l$ are k-wise independent. In particular, this means that for every $J \subseteq [m]$ with $|J| \leq k$, we have $b^l(J) = 0$, and therefore $b(J) = 0$ by (1).

Now, we bound $b^h(J)$ for a sufficiently large J. In particular, again assume that G is a random $[n, n\Delta, 3]$-graph. Then, because Q^h has algebraic degree 2, for every $\delta > 0$, [ABR16] show that with probability $1 - o(1)$ over the choice of G, for every J such that $|J| \geq \Delta^2 n^\delta$, we have

$$b^h(J) \leq \frac{1}{2}e^{-n^{\delta/2}/4} ,$$

and thus also $b(J) \leq e^{-n^{\delta/2}/4}$ for such J's.

Now, note that with probability $1 - o(1)$, we also must have a graph G for which both light and heavy tests have small biases. To conclude, we only need to verify that all cardinalities of J are covered, which is certainly true if $\Delta^2 n^\delta \leq k = \Omega(n/\Delta^2)$. This equivalently means that $n^{4\alpha} \leq \Omega(n^{1-\delta})$, which holds whenever $\alpha < \frac{1-\delta}{4}$, as assumed in the theorem statement. □

3.5 Hardness Amplification via the XOR Construction

In this paper, we rely on the assumption that $\mathbf{GF}^{G,P,\ell}$ is a good PRG for an appropriate family G of expanders. However, we want to add additional justification to our assumptions. Here, in particular, we discuss how weak unpredictability for graph-based block-wise local functions can be amplified to super-polynomially small unpredictability generically. This means in particular that block-wise local PRGs have strong self-amplifying properties, and that for any G and P, in order to invalidate our assumption, we need to find an attack which succeeds in predicting the next bit with large (i.e., polynomial) advantage over $\frac{1}{2}$. For otherwise, the lack of such an attack would imply that for the same G and (a related) P' and ℓ', $\mathbf{GF}^{G,P',\ell'}$ is a strong PRG.

To this end, we use a simple construction xoring the outputs of generators, which has already been studied to amplify PRG security [DIJK09, MT10] . Our analysis resembles the one from [DIJK09], but is given for completeness. Also, a more general construction, with xoring replaced by a general extractor, was considered by Applebaum [App12]. The use of xor, however, is instrumental to preserve block-wise locality. The main drawback of this construction is that it can *at best* ensure $2^{-\Omega(\log^{1+\theta}\lambda)}$ distinguishing gap for some $\theta \in (0,1]$ while retaining block size $\ell = O(\log \lambda)$. In Section 3.6, we explain a different approach which relies on a different assumption and only works for block-wise locality ≥ 3, but potentially guarantees $2^{-\lambda^{\Omega(1)}}$ distinguishing gap.

The XOR construction and its amplifying properties Let $\mathbf{UG} = \{\mathbf{UG}_\lambda\}_{\lambda\in\mathbb{N}}$ be an (s,δ)-UG, where $\mathbf{UG}_\lambda$ is a distribution on functions $\{0,1\}^{n(\lambda)} \to \{0,1\}^{m(\lambda)}$. For an additional parameter $k = k(\lambda) \geq 1$, we define the ensemble $\mathbf{UG}^k = \{\mathbf{UG}^k_\lambda\}_{\lambda\in\mathbb{N}}$, where $\mathbf{UG}^k_\lambda$ samples functions $\mathrm{UG}_1, \ldots, \mathrm{UG}_k \xleftarrow{\$} \mathbf{UG}_\lambda$ and output the description of a function $\mathrm{UG}^k : \{0,1\}^{n\times k} \to \{0,1\}^m$ which, on input $x = x^1 \parallel \cdots \parallel x^k$, where $x^i \in \{0,1\}^{n(\lambda)}$, outputs

$$\mathrm{UG}^k(x) = \mathrm{UG}_1(x^1) \oplus \cdots \oplus \mathrm{UG}_k(x^k) \ .$$

We show the following theorem, whose proof relies on Yao's XOR Lemma [Yao82, GNW11].

Theorem 5 (Security of the XOR Construction)**.** *If $\mathbf{UG}$ is a (s,δ)-UG and $k = k(\lambda)$ is polynomial in λ, then $\mathbf{UG}^k$ is a (s',ϵ)-PRG, where*

$$\epsilon(\lambda) \leq (2\delta(\lambda))^{k(\lambda)} \ , \quad s'(\lambda) = \Theta\left(\frac{\delta(\lambda)^{2k} \cdot s(r)}{k \log(k/\delta(\lambda))}\right) \ .$$

Proof. Let $A = \{A_\lambda\}_\lambda$ be a predictor family size s' such that for some $i = i(\lambda)$ guesses with probability

$$\pi_{A,i}(\lambda) = \Pr\left[\begin{array}{l} x \xleftarrow{\$} \{0,1\}^{n\times k} \\ \mathrm{UG}^k \xleftarrow{\$} \mathbf{UG}^k_\lambda \end{array} : A_\lambda(\mathrm{UG}^k, \mathrm{UG}^k_{\leq i}(x)) = \mathrm{UG}^k_i(x)\right] \ .$$

Now, we can build another adversary family $B = \{B_\lambda\}_{\lambda\in\mathbb{N}}$ which takes as input $\mathrm{UG}_1, \ldots, \mathrm{UG}_k$ from the range of $\mathbf{UG}_\lambda$, as well as $y_1 = \mathrm{UG}_{1,\leq i}(x^1), \ldots, y_k = \mathrm{UG}_{k,\leq i}(x^k)$ for $x_1, \ldots, x_k \xleftarrow{\$} \{0,1\}^n$, and outputs a bit

$$B_\lambda(\mathrm{UG}_1, \ldots, \mathrm{UG}_k, y_1, \ldots, y_k) = A_\lambda(\mathrm{UG}^k = (\mathrm{UG}_1, \ldots, \mathrm{UG}_k), y_1 \oplus \cdots \oplus y_k) \ ,$$

and thus, by construction,

$$\pi_{A,i}(\lambda) = \Pr\left[\begin{array}{c} \mathrm{UG}_1, \ldots \mathrm{UG}_k \xleftarrow{\$} \mathbf{UG}_\lambda \\ x^1, \ldots, x^r \xleftarrow{\$} \{0,1\}^n \\ y_j = \mathrm{UG}_{j,\leq i}(x^j) \text{ for } j = 1, \ldots, k \\ b = \mathrm{UG}_{1,i+1}(x^1) \oplus \cdots \oplus \mathrm{UG}_{k,i+1}(x^k) \end{array} \;:\; B_\lambda(\mathrm{UG}_1, \ldots, \mathrm{UG}_k, y_1, \ldots, y_k) = b\right] .$$

To continue, we rely on a (concrete) version of Yao's XOR Lemma, which we state here, with parameters obtained from Levin's proof. See [GNW11] for further details. (Note that g is not sampled from a distribution in the following statement, but this can be simulated by making g part of the randomness x. Also, for the uniform setting, there are no efficiency requirements on g and P.)

Theorem 6 (XOR Lemma). *Let $g : \{0,1\}^r \to \{0,1\}^*$ and $P : \{0,1\}^r \to \{0,1\}$ such that for all adversaries A with size $s(r)$ we have*

$$\Pr\left[x \xleftarrow{\$} \{0,1\}^r \;:\; A(g(x)) = P(x)\right] \leq \frac{1}{2} + \delta(r) .$$

Then, for all $k = k(r)$, all $\gamma = \gamma(r)$ and all adversaries B with size $s'(r)$,

$$\Pr\left[x_1, \ldots, x_r \xleftarrow{\$} \{0,1\}^r : B(g(x_1), \ldots, g(x_r)) = P(x_1) \oplus \cdots \oplus P(x_r)\right] \leq \frac{1}{2} + 2^{k-1}\delta^k + \gamma(r) , \quad (2)$$

where $s'(r) = \Theta\left(\frac{\gamma^2 \cdot s(r)}{k \cdot \log(k/\gamma)}\right)$.

Therefore, by the assumption in the theorem statement that UG is (s, δ)-UG, the statement follows by setting $\gamma = \delta^k$ and noting that $2^{k-1} + 1 \leq 2^k$ for all $k \geq 1$. $\square$

Block-wise local instantiation We instantiate the construction with parameter k when $\mathbf{UG} = \mathbf{GF}^{G,P,\ell}$ for a family of $[n, m, d]$-hypergraphs $G = \{G_\lambda\}_{\lambda \in \mathbb{N}}$, some $\ell = \ell(\lambda)$, and a family P of $(d \times \ell)$-bit predicates. Since the resulting function UG^k_λ uses k instances of the *same* function $\mathrm{GF}_{G_\lambda, P_\lambda, \ell}$, it can equivalently be thought as having the form (up to re-arranging the order of the input bits) $\mathrm{GF}_{G_\lambda, P^k_\lambda, \ell(\lambda)\cdot k(\lambda)}$, where the predicate P^k on input d $(k \cdot \ell)$-bit blocks $\mathbf{x}_1, \ldots, \mathbf{x}_d$, it interprets each of them as k ℓ-bit blocks $\mathbf{x}_i = \mathbf{x}_{i,1} \parallel \cdots \parallel \mathbf{x}_{i,k}$ and outputs

$$P^k(\mathbf{x}_1, \ldots, \mathbf{x}_d) = P(\mathbf{x}_{1,1}, \ldots, \mathbf{x}_{d,1}) \oplus \cdots \oplus P(\mathbf{x}_{k,1}, \ldots, \mathbf{x}_{k,d}) .$$

To instantiate our transformation, we assume that for some $\ell(\lambda) = \Omega(\log^{1-\theta}(\lambda))$ and a family of $[n(\lambda), m(\lambda), d]$-hypergraphs $G = \{G_\lambda\}_{\lambda \in \mathbb{N}}$, the function family $\mathrm{UG} = \mathbf{GF}^{G,P,\ell}$ is a $(s(\lambda) = 2^{\log^3(\lambda)}, \delta(\lambda) = \lambda^{-\Omega(1)})$-UG. Now, set $k(\lambda) = \log^\theta(\lambda)$. Then, UG^k is by the above $(d, O(\log(\lambda)))$-block-wise local, and it is also (s', ϵ)-UG for $s'(\lambda) = \mathrm{poly}(\lambda)$, and

$$\epsilon(\lambda) = (2\delta(\lambda))^{k(\lambda)} = 2^{-\Omega(\log^{1+\theta}(\lambda))} .$$

In other words, we have just established the following corollary.

Corollary 2. *For any $\beta > 0$, $d \geq 2$, and $\theta \in (0, 1]$, if the $(d, O(\log^{1-\theta}(\lambda)))$-$\mathsf{BLUG}(n, \beta, 2^{\log^3(\lambda)}, 1/\lambda^{\Omega(1)})$ assumption holds, then the assumption $(d, O(\log(\lambda)))$-$\mathsf{BLUG}(n, \beta, \mathrm{poly}(\lambda), 2^{-\Omega(\log^{1+\theta}(\lambda))})$ also holds true.*

3.6 The Extraction Construction

The XOR construction guarantees that finding a graph and predicate for which $\mathbf{GF}^{G,P,\ell}$ is even only mildly unpredictable (for slightly super-polynomial predictors) implies already a block-wise local with block size $O(\log(\lambda))$ and inverse super-polynomial distinguishing gap. However, note that sub-exponential distinguishing advantage while being $(d, O(\log(\lambda)))$-block-wise local is out of reach, as this require $k = O(\log(\lambda))$, which in turns can only gives us (even assuming an "ideal" version of the XOR Lemma) distinguishing advantage $\delta^{O(\log(\lambda))} = 2^{-O(\log^2\lambda)}$, since $\delta = 1/\lambda^{O(1)}$.

Here, we give a second construction of a block-wise local PRG with polynomial stretch $m = n^{1+\alpha}$ that uses an instantiation of $\mathbf{GF}^{G,P,\ell}$ which merely ensures its output has a (sufficient) amount of pseudo-min-entropy. We stress that we have no reason to focus on such an assumption other than the fact that this may appear easier to reach for a given graph and predicate. A drawback of our approach is that it requires the underlying base graph to have degree $d \geq 3$.

The extraction construction The following assumption is a weakening of the notion of a PRG, to a pseudo-min-entropy generator (PMEG). Recall that a random variable X has *min-entropy* k if every value is taken with probability at most 2^{-k}.

Definition 26 (Family of Pseudo-Min-Entropy Generators (PMEGs))**.** *Let n and $\Delta \leq m$ be polynomials, and $\mu : \mathbb{N} \to [0,1]$ be a function. A family of $(n(\lambda), m(\lambda), \mu(\lambda), \Delta(\lambda))$-PMEGs is an ensemble of distributions $\mathbf{PMEG} = \{\mathbf{PMEG}_\lambda\}$ satisfying the same syntax and efficiency requirements as a PRG, but moreover:*

μ-Indistinguishability: *There exists a family of distributions $\{X_\lambda\}_{\lambda\in\mathbb{N}}$ such that X_λ is over the $m(\lambda)$-bit strings,*

$$\left\{\mathrm{PMEG} \xleftarrow{\$} \mathbf{PMEG}_\lambda;\ s \xleftarrow{\$} \{0,1\}^{n(\lambda)} : (\mathrm{PMEG}, \mathrm{PMEG}(s))\right\}_{\lambda\in\mathbb{N}}$$
$$\approx_\mu \ \left\{\mathrm{PMEG} \xleftarrow{\$} \mathbf{PMEG}_\lambda;\ r \xleftarrow{\$} X_\lambda \ :\ (\mathrm{PRG}, r)\right\}_{\lambda\in\mathbb{N}}$$

and moreover, X_λ has min-entropy $m(\lambda) - \Delta(\lambda)$.

We use the following lemma by Dodis and Smith [DS05], which shows that xoring a sufficiently high-min-entropy source with the output of a small-bias generator yields uniform randomness.

Lemma 4. *Let $\mathrm{SB} : \{0,1\}^n \to \{0,1\}^m$ be an ϵ-biased generator and let X be a random variable taking values in $\{0,1\}^m$ with min-entropy at least $m - \Delta$ for some $\Delta \geq 0$. Then,*

$$\mathrm{SD}((\mathrm{SB}(U_n) \oplus X), U_m) \leq \frac{\epsilon}{\sqrt{2}} \cdot 2^{\Delta/2}\ ,$$

where U_k denotes the uniform distribution on k-bit strings, and SD denotes statistical distance.

If we use the instantiation of an ϵ-biased generator guaranteed to exist by Lemma 3 as $\mathrm{GF}_{G,Q,2}$, we see that for $m = n^{1+\alpha}$, with a small loss allowing notational simplifications, we can have $\delta = 2^{-n^{1-5\alpha}/4}$. In particular, the statistical distance in Lemma 4 remains sub-exponential as long as the entropy loss is $h = o(n^{1-5\alpha})$. In the following, for n and α, we denote by $\mathcal{G}_{n,\alpha,d}$ the set of $[n, n^{1+\alpha}, d]$-hypergraphs G for which $\mathrm{GF}_{G,Q,2}$ is a $2^{-n^{1-5\alpha}/4}$-biased generator. (Lemma 3 needs $d = 3$, but we can trivially extend the result to $d \geq 4$ by ignoring any extra input.) Lemma 3 implies that $\mathcal{G}_{n,\alpha,d}$ includes a $1 - o(1)$ fraction of $[n, n^{1+\alpha}, d]$-hypergraphs.

Now, let $G = \{G_\lambda\}_{\lambda\in\mathbb{N}}$ be a family of $[n(\lambda), n(\lambda)^{1+\alpha}, d]$-hypergraphs, with $d \geq 3$, and let $\ell(\lambda) = O(\log(\lambda))$. Moreover, let $P = \{P_\lambda\}_{\lambda\in\mathbb{N}}$ be a family of predicates on $(d \times \ell)$-bit strings. Also, for the predicate Q defines as above, we consider the family of predicates $P \oplus Q = \{R_\lambda\}_{\lambda\in\mathbb{N}}$, where the corresponding predicate R_λ for security parameter $\lambda \in \mathbb{N}$ outputs, on input d $(\ell(\lambda)+2)$-bit strings $\mathbf{x}_1, \ldots, \mathbf{x}_d$,

$$P_\lambda(\mathbf{x}_1[1\ldots\ell], \ldots, \mathbf{x}_d[1\ldots\ell]) \oplus Q(\mathbf{x}_1[\ell+1, \ell+2], \mathbf{x}_2[\ell+1, \ell+2], \mathbf{x}_3[\ell+1, \ell+2]) \ .$$

Then, note that evaluating $\mathrm{GF}_{G_\lambda, R_\lambda, \ell(\lambda)+2}$ on a random input, is the same as evaluating $\mathrm{GF}_{G_\lambda, P_\lambda, \ell(\lambda)}$ and $\mathrm{GF}_{G_\lambda, Q, 2}$ on independent random inputs, and xoring the results. This means in particular that by Lemma 4, if the graph family G yields a small-bias generator, and $\mathbf{GF}^{G,P,\ell}$'s output has enough computational min-entropy, then we obtain a good PRG. This is summarized by the following lemma.

Lemma 5. *Let $d \geq 3$, let $n(\lambda)$ be a polynomial, $\ell(\lambda) = O(\log(\lambda))$, and let $\alpha > 0$. If there exists a family $G = \{G_\lambda\}_{\lambda\in\mathbb{N}}$ of $[n(\lambda), n(\lambda)^{1+\alpha}, d]$-hypergraphs, and a family $P = \{P_\lambda\}_{\lambda\in\mathbb{N}}$ of predicates on $(d \cdot \ell)$-bit strings such that:*

1. *$G_\lambda \in \mathcal{G}_{n(\lambda),\alpha,d}$, and*
2. *$\mathbf{GF}^{G,P,\ell}$ is a $(n(\lambda), n(\lambda)^{1+\alpha}, \nu(\lambda), \Delta(\lambda))$-PMEG for $\Delta(\lambda) = o(n(\lambda)^{1-5\alpha})$.*

Then, $\mathbf{GF}^{G,P\oplus Q,\ell+2}$ is a PRG with sub-exponential distinguishing gap.

4 IO from Block-Wise Locality-$(L, \log\lambda)$ PRG and L-Linear Maps

In this section, we prove the following bootstrapping theorem.

Theorem 7 (Bootstrapping via *block-wise local* PRGs). *Let $\mathcal{R} = \{\mathcal{R}_\lambda\}$ be any family of rings, ε be any positive constant, L any positive integer, n any sufficiently large polynomial, and $\mathrm{i}\ell$ and κ any polynomials. There is a construction of $\mathrm{i}\ell(\lambda)$-bit-input IO for* P/poly*, from the following primitives:*

- *A family of $(n(\lambda) \times \log\lambda, n(\lambda)^{1+\varepsilon})$-PRGs with* block-wise locality $(L, \log\lambda)$.
- *A* public-key *FE for degree-L polynomials in $\mathcal{R}$, with linear efficiency and* Full-Sel-*security; or with a secret-key FE with the same properties, assuming additionally the $(2^{-\mathrm{i}\ell(\lambda)+\kappa(\lambda)}\,\mathrm{negl}(\lambda))$-hardness of LWE with subexponential modulo-to-noise ratio.*

If the PRG and FE schemes are $(2^{-\mathrm{i}\ell(\lambda)+\kappa(\lambda)}\,\mathrm{negl}(\lambda))$-secure, then the resulting IO scheme is $(2^{-\kappa(\lambda)}\,\mathrm{negl}(\lambda))$-secure.

Theorem 7 follows the same approach as Lin's recent bootstrapping theorem [Lin16b], but modifies it in two ways. First, it uses block-wise local PRGs to replace local PRGs. Second, it makes explicit the relation between the *security level* (more precisely, the maximal distinguishing gap) of the underlying PRG and FE, and the *input-length and security level* of the resulting IO — if the underlying primitives are $2^{-\mathrm{i}\ell+\kappa}\,\mathrm{negl}$-secure, then the resulting IO scheme is for $\mathrm{i}\ell$-bit-input circuits and $2^{\kappa}\,\mathrm{negl}$-security. Such relations are implicit in previous works, and more importantly, not as *tight* as shown here. In particular, to obtain the same IO scheme, previous works require the underlying primitives to be $2^{-O(\mathrm{i}\ell^2)+\kappa}\,\mathrm{negl}$-secure [AJ15, BV15] or $2^{-O(\log\lambda)\mathrm{i}\ell+\kappa}\,\mathrm{negl}$-secure [LPST16].

Overview of Proof of Theorem 7 To show the theorem, similar to previous works [LV16, Lin16b], we take two steps:

Step 1 Construct a single-key public-key FE schemes $\mathbf{CFE} = \{\mathbf{CFE}^{N,D,S}\}$ for P/poly, with $(1-\varepsilon)$-sublinear compactness and $2^{-\mathrm{i}\ell+\kappa}$ negl-Full-Sel-security, starting from a FE for degree-L polynomials in $\mathcal{R}$, with linear efficiency and Full-Sel-security.

Previously, the work of [LV16] showed how to achieve this transformation from a locality-L PRGs and FE for computing degree $3L+2$ polynomials. Following that, the two recent works of [Lin16b, AS16] used a *pre-processing technique* to relax the requirement on the underlying FE to supporting only degree-L polynomials. In this work, we extend their *pre-processing technique* even further, in order to relax the requirement on the underlying PRGs from having locality L to having *block-wise* locality $(L, \log\lambda)$.

In particular, our approach is a fairly straightforward extension of [Lin16b], albeit notationally tedious, once we make the following observation. For any block-wise locality L function $\mathrm{PRG}(\mathbf{x})$ with input $\mathbf{x}$ of dimension $\ell \times n$, one can compute the PRG as a degree L function in an arbitrary ring, provided *all* the monomials defined on the variables of each ℓ-bit column are pre-computed.

More concretely, let PRG_i denote the function that computes the i^{th} output bit of PRG. We know that PRG_i depends on L input columns, $\mathbf{x}_{i_1}, \cdots \mathbf{x}_{i_L}$, each of size ℓ bits. We can arithmetize PRG_i and write it as a sum of multilinear monomials over the bits in these input columns (recall that all computations are taking place in a corresponding *ring*, so all expressions are within this ring) – in particular, with $\mathcal{M}_i$ being the set of monomials appearing in the computation of PRG_i,

$$\mathrm{PRG}_i(\mathbf{x}) = \sum_{M \in \mathcal{M}_i} M(\mathbf{x}_{i_1}, \cdots \mathbf{x}_{i_L}) \ .$$

Furthermore, every monomial $M(\mathbf{x}_{i_1}, \cdots, \mathbf{x}_{i_L})$ can be written as the product of L monomials $M_1, \ldots, M_L$, defined over each of the columns,

$$\mathrm{PRG}_i(\mathbf{x}) = \sum_{M \in \mathcal{M}_i} M_1(\mathbf{x}_{i_1}) \times \cdots \times M_L(\mathbf{x}_{i_L})$$

Now suppose that we have pre-computed *all possible* monomials over every column. In particular, let $\mathsf{Mnml}(\mathbf{x}_i)$ denote the set of all 2^ℓ multilinear monomials over bits in $\mathbf{x}_i$. Then $\mathrm{PRG}_i(\mathbf{x})$ can be computed in degree L, because there exists a degree-L function PRG'_i such that

$$\mathrm{PRG}'_i(\mathsf{Mnml}(\mathbf{x}_1), \cdots, \mathsf{Mnml}(\mathbf{x}_n)) = \mathrm{PRG}_i(\mathbf{x}) \ .$$

Given this, we can have a way of computing PRG'_i using the underlying FE for degree L polynomials. Also, because we have $\ell(\lambda) = O(\log\lambda)$, the domain size of PRG has now increased only by a $\mathrm{poly}(\lambda)$ multiplicative factor when transforming it into PRG'. While this is the main idea, some care must be taken to ensure that this trick fits together with the rest of the preprocessing in [Lin16b]. We describe this step in full detail in Section 4.1 below.

In the case that the underlying FE scheme is a secret-key one, rather than a public-key one, we can follow the same approach obtain first a single-key weakly-compact secret-key FE scheme for P/poly with the same security level as described above. Then, we invoke the result of [BNPW16] that shows such secret-key FE schemes can be transformed into public key FE schemes with the same properties, assuming the existence of single-key *fully-compact* public-key FE schemes for Boolean NC^1 (*i.e.*, NC^1 circuits with a single output bit). As shown by [GKP+13], the latter can be constructed assuming the hardness of LWE where the modulus-to-noise ratio is subexponential.[2]

We note that our transformation from FE for low-degree computations to weakly-compact FE for P/poly in Section 4.1 incurs only a polynoimal security loss, and so does the transformation of [BNPW16]. Therefore, the resulting weakly-compact FE has essentially the same level of security as that of underlying primitives.

Step 2. Apply an FE-to-IO transformation to obtain $\mathrm{i}\ell$-bit-input IO for P/poly, with $2^{-\kappa}\,\mathrm{negl}$-security.

The literature already offers three FE-to-IO transformations [BV15, AJ15, LPST16] that start from a public key FE scheme $\mathbf{CFE} = \{\mathbf{CFE}^{N,D,S}\}$ as described above w.r.t. any positive constant ε. However, none of their analysis is sufficiently tight for our purpose: The transformations of [BV15, AJ15] need the underlying FE scheme to be $2^{-O(\mathrm{i}\ell^2)+\kappa}\,\mathrm{negl}$-secure, and that of [LPST16] need the underlying FE scheme to be $2^{-O(\log\lambda)\mathrm{i}\ell+\kappa}\,\mathrm{negl}$-secure.

In contrast, here, we want to start with $2^{-\mathrm{i}\ell+\kappa}\,\mathrm{negl}$-secure FE. To do so, we present a new FE-to-IO transformation inspired by that of [LPST16] and present a tight analysis. We describe this step below in Section 4.2.

4.1 Step 1: Constructing Weakly-Compact FE

Proposition 1. *Let $\mathcal{R}$, ε, L, and n be defined as in Theorem 7, and $\bar{\kappa}$ be any polynomial. There is a construction of 1-key weakly-compact public-key FE for* P/poly *from the following primitives:*

- *A family of $(n(\lambda)\times\log\lambda, n(\lambda)^{1+\varepsilon})$-PRGs with* block-wise locality $(L, \log\lambda)$.
- Public-key *FE for degree-L polynomials in $\mathcal{R}$, with linear efficiency and* Full-Sel-*security.*

*If the underlying PRG and FE are $(2^{-\bar{\kappa}(\lambda)}\,\mathrm{negl}(\lambda))$-secure, then, the resulting weakly-compact FE is $(2^{-\bar{\kappa}(\lambda)}\,\mathrm{negl}(\lambda))$-*Full-Sel-*secure.*

Moreover, the public-key FE for degree-L polynomials can be replaced with secret-key FE with the same properties, assuming additionally the $(2^{-\bar{\kappa}(\lambda)}\,\mathrm{negl}(\lambda))$-hardness of LWE with subexponential modolus-to-noise ratio.

It was shown in [Lin16b] that 1-key weakly-compact FE for P/poly can be constructed from locality-L PRG and (unbounded collusion) FE for degree-L polynomials. Their construction of weakly-compact FE follows from the blue-print of previous works [Lin16a, LV16], which uses FE for low degree polynomials to compute a randomized encoding of a computation in P/poly, with pseudo-randomness generated through a local PRG. The locality of RE and PRG ensures that their composition can be computed in low degree. However, the straightforward composition of RE and PRG leads to a computation with degree $3L+2$. The key idea in [Lin16b] and the concurrent work of [AS16] is that part of the RE computation can already be done at encryption time, that is, by asking the encryptor to pre-process the inputs (of the computation in P/poly) and seeds of PRG, and encrypt the pre-processed values, the composition of RE and PRG can be computed in just degree L from the pre-processed values, at decryption time — This is called the *preprocessing technique*. We take this technique one step further: By also performing part of the PRG comptuation at encryption time, we can replace local PRG with block-wise local PRG (with appropriate parameters) at "no cost". This might help us to circumvent the lowerbound on the locality of PRGs (*i.e.*, locality-4 PRG does not exist).

[2]More precisely, the work of [GKP+13] constructed a single-key fully-compact public-key FE scheme for Boolean depth-d circuits, assuming the hardness of LWE where the modulus to noise ratio is $\lambda^{O(d)}$. Since we only need such FE schemes for Boolean NC^1 circuits, it, in fact, suffices to assume the hardness of LWE with quasipolynomial modulus-to-noise ratio, which is clearly implied by the hardness of LWE with subexponential modulus-to-noise ratio.

Below, we first briefly review the blueprint of [LV16], then describe the pre-processing idea of [Lin16b] and how to use it to accommodate PRG with block-wise locality.

The General Blueprint of [LV16] To construct 1-key weakly-compact FE for $\mathsf{P/poly}$, Lin and Vaikuntanathan [LV16] (LV) first observed that, using the Trojan Method [CIJ+13], it suffices to construct 1-key weakly-compact FE for NC^1 functions with some fixed depth $D(\lambda) = O(\log \lambda)$; denote this class of functions as NC^1_D.

Next, to bootstrap a low-degree FE scheme to FE for NC^1_D, the idea is using randomized encoding to "compress" any function $h(\mathbf{x}) \in \mathsf{NC}^1_D$ into a function $g(\mathbf{x}, \mathbf{s}) = \mathsf{REnc}(f, \mathbf{x}\ ;\ \mathrm{PRG}(\mathbf{s}))$ with small degree in $\mathcal{R}$. The reason that local PRG is used is that the locality of a Boolean function bounds the degree of computing this function in any ring. Then, plugging-in randomized encodings with small locality like that of [AIK04] the overall degree of g is small.

Below, we formally describe the LV construction of FE for NC^1_D. We focus on the public key case. (The secret key case is handled in the same way, with an additional step of applying the transformation of [BNPW16] in the end; see the discussion in paragraph "Step 1" above.) Their FE scheme $\mathbf{CFE}^{N,D,S}$ for NC^1 circuits with input-length $N = N(\lambda)$, depth $D = D(\lambda)$, and size $S = S(\lambda)$, uses the following tools: Let $\mathcal{R}$ be a family of rings.

- A $(n, n^{1+\alpha})$-pseudorandom generator PRG with locality L, for a sufficiently large polynomial input length $n = n(\lambda)$ and any positive constant α.
- Full-Sel-secure (collusion resistant) FE schemes for degree-$(3L+2)$ polynomials in $\mathcal{R}$, $\{\mathbf{FE}^{N'} = (\mathsf{FE.Setup}, \mathsf{FE.KeyGen}, \mathsf{FE.Enc}, \mathsf{FE.Dec})\}$, with linear efficiency.
- The AIK randomized encoding scheme in NC^0 [AIK04]; denote the encoding algorithm as $\mathsf{AIK}(f, \mathbf{x}\ ;\ \mathbf{r})$.

The scheme $\mathbf{CFE}^{N,D,S} = (\mathsf{CFE.Setup}, \mathsf{CFE.KeyGen}, \mathsf{CFE.Enc}, \mathsf{CFE.Dec})$ is described in Figure 2.

We refer the reader to [LV16] for the correctness and security of the scheme. The compactness of the scheme $\mathbf{CFE}$ follows from the following two facts:

1. The length of the input $(\mathbf{x}, \mathbf{s}, \mathbf{s}', 0)$ encrypted using $\mathbf{FE}$ is $N+2\Gamma+1 = N+S(\lambda)^{1/(1+\alpha)}\,\mathrm{poly}(\lambda)$.
2. $\mathbf{FE}$ has linear efficiency.

Putting them together, we have,

$$\begin{aligned}\mathsf{Time}_{\mathsf{CFE.Enc}}(\mathsf{MPK}, \mathbf{x}) &= \mathsf{Time}_{\mathsf{FE.Enc}}(\mathsf{MPK}, (\mathbf{x}, \mathbf{s}, \mathbf{s}', 0)) \\ &= \mathrm{poly}(\lambda)|(\mathbf{x}, \mathbf{s}, \mathbf{s}', 0)| = S(\lambda)^{1/(1+\alpha)}\,\mathrm{poly}(\lambda, N)\end{aligned}$$

which is sublinear in the function size as desired. Furthermore, to see why degree-$(3L+2)$ FE suffices for the construction, note that the construction uses the underlying FE to generate keys computing the function g in Figure 2, and hence it suffices to argue that g can be computed in degree $3L+2$. By definition of g, when $b = 1$, the output can be computed in degree L as the PRG can be computed in degree L in $\mathcal{R}$ (XOR with $\mathbf{CT}$ does not incur additional degree as $\mathbf{CT}$ are constants hardwired in the function g); when $b = 0$, the output can be computed in degree $3L+1$, since the AIK randomized encoding has degree 3 in the random bits (*i.e.* PRG output) and 1 in the input $\mathbf{x}$. Therefore, g has exactly degree $3L+2$, as selection by b can be done with one multiplication.

Single-key Compact FE Scheme CFE by [LV16]

<u>SETUP:</u> $\mathsf{CFE.Setup}(1^\lambda)$ samples $(\mathsf{MPK}, \mathsf{MSK}) \xleftarrow{\$} \mathsf{FE.Setup}(1^\lambda)$.

<u>ENCRYPTION:</u> $\mathsf{CFE.Enc}(\mathsf{MPK}, \mathbf{x})$ samples $\mathbf{s}, \mathbf{s}' \xleftarrow{\$} \{0,1\}^\Gamma$ for $\Gamma = S^{1/1+\alpha}\,\mathrm{poly}(\lambda)$, and generates

$$\mathsf{CT} \xleftarrow{\$} \mathsf{FE.Enc}(\mathsf{MPK}, (\mathbf{x}, \mathbf{s}, \mathbf{s}', 0))$$

<u>KEY GENERATION:</u> $\mathsf{CFE.KeyGen}(\mathsf{MSK}, h)$ does the following:

- Sample $\mathbf{CT} \xleftarrow{\$} \{0,1\}^\ell$, where ℓ is set below.
- Define function g as follows: On input $\mathbf{x}$ of length N, two PRG seeds $\mathbf{s}, \mathbf{s}'$ each of length Γ, and a bit b,

  $g(\mathbf{x}, \mathbf{s}, \mathbf{s}', b)$ does the following:

  - For every $i \in [S]$, let $h_i(\mathbf{x})$ denote the function that computes the i^{th} output bit of $h(\mathbf{x})$. Since $h \in \mathsf{NC}^1_D$, h_i has depth $D(\lambda) = O(\log \lambda)$ and size $2^{D(\lambda)} = \mathrm{poly}(\lambda)$.
  - If $b = 0$, compute $\mathbf{r} = \mathrm{PRG}(\mathbf{s})$, whose output has length $\Gamma^{1+\alpha} = S\,\mathrm{poly}(\lambda)$; divide the output into S equally long portions and denote by $\mathbf{r}[i]$ the i^{th} portion.
    For every $i \in [S]$, compute the AIK encoding $\Pi[i]$ of computation $(h_i, \mathbf{x})$ as follows:

    $$\forall\, i \in [S], \qquad \Pi[i] = \mathsf{AIK}(h_i,\ \mathbf{x}\ ;\ \ \mathbf{r}[i])\ .$$

    Output $\Pi = \{\Pi[i]\}_i$; set $\ell = |\Pi|$.
  - If $b = 1$, output $\Pi = \mathbf{CT} \oplus \mathrm{PRG}(\mathbf{s}')$.
- For every $l \in [\ell]$, generate a secret key $\mathsf{SK}_l \xleftarrow{\$} \mathsf{FE.KeyGen}(\mathsf{MSK}, g_l)$ for g_l that computes the l^{th} output bit of g.

Output $\mathsf{SK} = \{\mathsf{SK}_l\}_{l \in [\ell]}$.

<u>DECRYPTION:</u> $\mathsf{CFE.Dec}(\mathsf{SK}, \mathsf{CT})$ computes $\Pi = \{\mathsf{FE.Dec}(\mathsf{SK}_l, \mathsf{CT})\}_{l\in[\ell]}$, parses $\Pi = \{\Pi[i]\}$, and decodes every $\Pi[i]$ using the AIK decoding algorithm to obtain the output $h(\mathbf{x})$.

Figure 2: Single-key Compact FE **CFE** by [LV16]

The Idea of Preprocessing in [Lin16b] Towards reducing the degree of the underlying FE and accommodating PRGs with block-wise locality-$(L, \log\lambda)$, the idea is letting the encryptor preprocess the input $(\mathbf{x}, \mathbf{s}, \mathbf{s}', b)$ to produce certain intermediate values, from which the output of function g can be computed in exactly degree L. To see this, the output of g is viewed as corresponding to S AIK randomized encodings for functions $\{h_i\}_{i\in[S]}$. If the l^{th} output bit belongs to the i^{th} randomized encoding for h_i with random tape $\mathbf{r}[i]$, the function g_l computing it can be written as a sum of monomials as follows:

$$\begin{aligned} g_l(\mathbf{x}, \mathbf{s}, \mathbf{s}', b) &= (1-b)g_{l0}(\mathbf{x}, \mathbf{s}) + b g_{l1}(\mathbf{s}') \\ &= (1-b)\sum_{i_0,i_1,i_2,i_3} c_{i_0,i_1,i_2,i_3}\mathbf{x}_{i_0}\mathbf{r}[i]_{i_1}\mathbf{r}[i]_{i_2}\mathbf{r}[i]_{i_3} + b\sum_j c_j\mathbf{r}'_j \end{aligned} \tag{3}$$

where $\mathbf{r}[i]$ is the i^{th} portion in $\mathbf{r} = \mathrm{PRG}(\mathbf{s})$, and $\mathbf{r}' = \mathrm{PRG}(\mathbf{s}')$. This is because in the case of $b = 0$, the output is a bit in the AIK encoding of h_i and hence has degree 1 in the input $\mathbf{x}$ and degree 3 in $\mathbf{r}[i]$, while in the case of $b = 1$, the output has degree 1 in $\mathbf{r}'$.

When PRG has locality L, the straightforward way of computing a degree-3 monomial $\mathbf{r}[i]_{i_1}\mathbf{r}[i]_{i_2}\mathbf{r}[i]_{i_3}$ from the seed $\mathbf{s}$ requires degree $3L$. The works of [Lin16b, AS16] showed how to reduce the degree to just L. First, they use a different way to compute each $\mathbf{r}[i]$. View the seed $\mathbf{s}$ as a $Q \times \Gamma'$ matrix with $Q = Q(\lambda) = \mathrm{poly}(\lambda)$ rows and $\Gamma' = S^{1/1+\alpha}$ columns; apply PRG on each row of $\mathbf{s}$ to expand the seed matrix into a $Q \times S$ matrix $\mathbf{r}$ of pseudo-random bits. That is, denote the q^{th} row of $\mathbf{s}$ and $\mathbf{r}$ as $\mathbf{s}_q$ and $\mathbf{r}_q$; $\mathbf{r}_q = \mathrm{PRG}(\mathbf{s}_q)$. Finally, set the random tape for computing the i^{th} AIK encoding to be the i^{th} column $\mathbf{r}[i]$ of $\mathbf{r}$.

In [Lin16b], they used PRGs with locality L. Let $\mathrm{PRG}[i]$ denote the function computing the i^{th} output bit of PRG, and let $\mathsf{Nbr}(i) = \{\gamma_1, \cdots, \gamma_L\}$ be the indexes of the L seed bits that the i^{th} output bit depends on. Therefore,

$$\begin{aligned} \mathbf{r}[i]_{i_1}\mathbf{r}[i]_{i_2}\mathbf{r}[i]_{i_3} &= \mathrm{PRG}[i](\mathbf{s}_{i_1})\ \mathrm{PRG}[i](\mathbf{s}_{i_2})\ \mathrm{PRG}[i](\mathbf{s}_{i_3}) \\ &= \sum_{\substack{\text{Monomials} \\ X,Y,Z \text{ in } \mathrm{PRG}[i]}} \left(\begin{array}{cccc} & X(s_{i_1,\gamma_1}, & \cdots, & s_{i_1,\gamma_L}) \\ \times & Y(s_{i_2,\gamma_1}, & \cdots, & s_{i_2,\gamma_L}) \\ \times & Z(s_{i_3,\gamma_1}, & \cdots, & s_{i_3,\gamma_L}) \end{array} \right). \end{aligned} \tag{4}$$

Suppose that one has pre-computed all degree ≤ 3 monomials over bits in each column $\mathbf{s}[\gamma]$ of $\mathbf{s}$.

$$\text{Define} \qquad \mathsf{Mnml}^{\leq 3}(A) := \{a_i a_j a_k \mid a_i, a_j, a_k \in A \cup \{1\}\}$$

Given $\mathsf{Mnml}^{\leq 3}(\mathbf{s}[\gamma])$ for every $\gamma \in \mathsf{Nbr}(i)$, one can compute $\mathbf{r}[i]_{i_1}\mathbf{r}[i]_{i_2}\mathbf{r}[i]_{i_3}$ in Equation (4) using just degree L. Similarly, given $\mathsf{Mnml}^{\leq 3}(\mathbf{s}[\gamma])$ for all $\gamma \in [\Gamma']$, one can compute *any* degree 3 monomials over bits in $\mathbf{r}[i]$ for *any* i, sufficient for the computation of g.

Furthermore, the size of each set $\mathsf{Mnml}^{\leq 3}(\mathbf{s}[\gamma])$ is bounded by $(Q+1)^3 = \mathrm{poly}(\lambda)$, and thus the size of their union for all γ is bounded by $\Gamma' \mathrm{poly}(\lambda) = S^{1/1+\alpha}\mathrm{poly}(\lambda)$ — only a polynomial factor (in λ) larger than the original seed $\mathbf{s}$ itself. Therefore the encryptor can afford to precompute all these monomials and encrypt them, without compromising the weak-compactness of the resulting FE for NC^1_D scheme.

This Work: Handling Block-Wise Local PRG. Our new observation is that the above technique naturally extends to accommodate block-wise local PRGs. Consider a family of $(n(\lambda) \times \log\lambda, n(\lambda)^{1+\alpha})$-PRGs with block-wise locality-$(L, \log\lambda)$. For ease of exposition, we think of the seed of such PRGs as a vector $\mathbf{t}$ of length n, where every element t_i is, instead of a single bit, a block of $\log \lambda$ bits; each output bit $\mathrm{PRG}[i](\mathbf{t})$ thus depends on at most L blocks.

Correspondingly, think of the seed matrix $\mathbf{s}$ described above as consisting of $Q \times \Gamma'$ blocks of $\log \lambda$ bits. When $\mathbf{r}[i]$ is computed using block-wise local PRGs, the degree-3 monomial $\mathbf{r}[i]_{i_1}\mathbf{r}[i]_{i_2}\mathbf{r}[i]_{i_3}$ in Equation (4) now depends on a set of blocks $\{s_{i_t,\gamma_s}\}_{t\in[3],s\in[L]}$. Though the actual locality of the PRG is $L \log \lambda$, due to its special structure, we can still pre-process the seed $\mathbf{s}$ to enable computing any degree-3 monomial over $\mathbf{r}[i]$ for any i using degree L.

- First, precompute all *multilinear* monomials over bits in each block $s_{q,\gamma}$ in $\mathbf{s}$.

  $$\text{Define} \qquad \mathsf{Mnml}(A) := \left\{ a_{i_1} a_{i_2} \cdots a_{i_q} \mid q \leq |A| \text{ and } \forall j,k\ a_{i_j} \neq a_{i_k} \in A \right\} \ .$$

  More precisely, precompute $\mathsf{Mnml}(s_{q,\gamma})$ for all $q \in [Q]$ and $\gamma \in [\Gamma']$. Note that each set $\mathsf{Mnml}(s_{q,\gamma})$ has exactly size λ.

- Second, for every column $\gamma \in [\Gamma']$, take the union of monomials over blocks in column γ, that is, $\cup_q \mathsf{Mnml}(s_{q,\gamma})$. Then, precompute all degree-≤ 3 monomials over this union, that is, $\mathsf{Mnml}^{\leq 3}(\cup_q \mathsf{Mnml}(s_{q,\gamma}))$, for each γ. Observe that from $\left\{\mathsf{Mnml}^{\leq 3}(\cup_q \mathsf{Mnml}(s_{q,\gamma}))\right\}_{\gamma\in[\Gamma']}$, one can again compute *any* degree-3 monomial in $\mathbf{r}[i]$ for *any* i in just degree L.

Furthermore, since $|\mathsf{Mnml}(s_{q,\gamma})| = \lambda$ for any q, γ, the number of monomials in $\mathsf{Mnml}^{\leq 3}(\cup_q \mathsf{Mnml}(s_{q,\gamma}))$ is bounded by $(Q\lambda + 1)^3 = \mathrm{poly}(\lambda)$. Therefore, the total size of pre-computed monomials is

$$\left| \left\{\mathsf{Mnml}^{\leq 3}(\cup_q \mathsf{Mnml}(s_{q,\gamma}))\right\}_{\gamma\in[\Gamma']} \right| \leq \Gamma' \,\mathrm{poly}(\lambda) = S^{1/1+\alpha}\,\mathrm{poly}(\lambda)\ , \tag{5}$$

which is still sublinear in the circuit size S and does not compromise the weak-compactness of the resulting FE for NC^1_D scheme.

Putting Things Together So far, we showed how to "compress" the computation of degree 3 monomials over $\mathbf{r}[i]$, for any i, into a degree-L computation. To compute function g in Equation (3) in degree L, we need to additionally pre-compute multiplications with $\mathbf{x}$ and b. As described in [Lin16b], this can be done easily by pre-computing the following:

$$\mathbf{V}_1 = \left\{\mathsf{Mnml}^{\leq 3}\left(\cup_q \mathsf{Mnml}(s_{q,\gamma})\right)\right\}_{\gamma\in[\Gamma']} \otimes (\mathbf{x}||b||1)$$

(where the sets of monomials are first interpreted as a vector before taking tensor product.) Given the tensor product, one can compute *any* monomial with degree ≤ 3 in $\mathbf{r}[i]$ for any i, degree ≤ 1 in $\mathbf{x}$, and degree ≤ 1 in b, in just degree L, which is sufficient for computing the first additive term in g_l in Equation (3). Similarly, to compute the second additive term in g_l, it suffices to precompute all multilinear monomials over every block in $\mathbf{s}'$ (of length Γ), and compute their tensor product with $b||1$, that is,

$$\mathbf{V}_2 = \left\{\mathsf{Mnml}(s'_\gamma)\right\}_{\gamma\in[\Gamma]} \otimes (b||1)$$

In summary, for every $l \in [\ell]$, there exists a degree-L polynomial P_l that on input $(\mathbf{V}_1, \mathbf{V}_2)$ outputs $g_l(\mathbf{x}, \mathbf{s}, \mathbf{s}', b)$.

$$\text{Define} \qquad P_l := \text{the degree-}L\text{ polynomial s.t. } P_l(\mathbf{V}_1, \mathbf{V}_2) = g_l(\mathbf{x}, \mathbf{s}, \mathbf{s}', b) \tag{6}$$

Moreover, we show that both $\mathbf{V}_1$ and $\mathbf{V}_2$ have length sublinear in the circuit size. First, combining Equation (5) with the fact that $|(\mathbf{x}||b||1)| = N + 2$, we have that

$$|\mathbf{V}_1| \le S^{1/1+\alpha}\,\mathrm{poly}(\lambda) \times (N+2) = S^{1/1+\alpha}\,\mathrm{poly}(\lambda, N)\;. \tag{7}$$

The size of $\mathbf{V}_2$ is

$$|\mathbf{V}_2| = \lambda \times \Gamma \times 2 \le S^{1/1+\alpha}\,\mathrm{poly}(\lambda)\;. \tag{8}$$

Finally, to construct a 1-key weakly-compact FE scheme for NC^1_D from FE for just degree L polynomials. We modify the LV construction as follows: 1) Instead of encrypting $(\mathbf{x}, \mathrm{s}, \mathrm{s}', b)$, the encryptor pre-computes and encrypts $\mathbf{V}_1||\mathbf{V}_2$ as described above, and 2) instead of generating secret keys for functions $\{g_l\}_{l\in[\ell]}$ which have degree $3L + 2$, generate secret keys for $\{P_l\}_{l\in[\ell]}$ which have only degree L. This way, at decryption time, the decryptor computes the correct output $\{P_l(\mathbf{V}_1||\mathbf{V}_2) = g_l(\mathbf{x}, \mathrm{s}, \mathrm{s}', b)\}$. The resulting new compact FE scheme **CFE** is described in Figure 3 (with key difference from the LV scheme highlighted). The compactness of the new scheme follows directly from the fact that the encrypted input $\mathbf{V}_1, \mathbf{V}_2$ have length sublinear in $S(\lambda)$, and that the degree-L **FE** scheme has linear efficiency. Moreover, its correctness and security follows from the same proof as that in [LV16]; since their security proof incur only a polynomial security loss, we conclude Theorem 7.

4.2 Step 2: Tight Construction of IO from Weakly-Compact FE

Proposition 2. *Let* $\mathrm{i}\ell$ *and* κ *be defined as in Theorem 7. Assume the existence of 1-key weakly-compact public-key FE for* $\mathsf{P/poly}$*, with* $2^{-\mathrm{i}\ell(\lambda)+\kappa(\lambda)}\,\mathrm{negl}(\lambda)$*-security. Then, there exists* $\mathrm{i}\ell(\lambda)$*-bit-input IO for* $\mathsf{P/poly}$*, with* $2^{-\kappa(\lambda)}\,\mathrm{negl}(\lambda)$*-security.*

Using weakly-compact public key FE for $\mathsf{P/poly}$, we construct IO for $\mathsf{P/poly}$ using induction over the input-length of the circuits that can be obfuscated. That is, in the $\mathrm{i}\ell^{\text{th}}$ induction step, we define the behavior of the scheme when obfuscating circuits with input-length $\mathrm{i}\ell$, by recusively invoking the scheme itself for obfuscating circuits with input length $\mathrm{i}\ell - 1$ and utilizing the weakly compact FE scheme. In fact, for the induction to work out, we need to use a different "interface" for the induction hypothesis, namely, instead of inductively defining an indistinguishability obfuscator in the plain model, we inductively define an indistinguishability obfuscator in the CRS model. Below, we start with defining this notion.

4.2.1 IO in the CRS model

Definition 27. *An indistinguishability obfuscator in the CRS model for a class of circuits* $\{\mathcal{C}_\lambda\}_{\lambda\in\mathbb{N}}$ *is a tuple of uniform* PPT *machines* $(\mathsf{SetupIO}, i\mathcal{O}, \mathsf{EvalIO})$ *satisfying the following conditions.*

Syntax: *The algorithms runs as follows:*

- *Setup:* $\mathsf{SetupIO}$ *on input a security parameter* 1^λ*, outputs a reference string* σ*, and a public parameter* pp*.*
- *Obfuscation with public parameter:* $i\mathcal{O}$ *on input a public parameter* pp *and a circuit* $C \in \mathcal{C}_\lambda$*, outputs an obfuscated circuit* $\hat{C}$*.*
- *Evaluation with the reference string:* EvalIO *on input an obfuscated circuit* $\hat{C}$*, an input* x*, and the reference string* σ*, outputs a string* y*.*

Single-key Compact FE Scheme CFE from block-wise locality-L PRG and degree-L FE

SETUP: $\mathsf{CFE.Setup}(1^\lambda)$ samples $(\mathsf{MPK}, \mathsf{MSK}) \xleftarrow{\$} \mathsf{FE.Setup}(1^\lambda)$, and $\mathrm{PRG} \xleftarrow{\$} \mathrm{PRG}_\lambda$.

ENCRYPTION: $\mathsf{CFE.Enc}(\mathsf{MPK}, \mathbf{x})$ samples

- a PRG seed $\mathbf{s}$ viewed as a $Q \times \Gamma'$ matrix for $Q = \mathrm{poly}(\lambda)$ and $\Gamma' = S^{1/1+\alpha}$, where each element $s_{q,\gamma}$ in $\mathbf{s}$ is a block of $\log \lambda$ bits, and
- another PRG seed $\mathbf{s}'$ viewed as a vector of length $\Gamma = S^{1/1+\alpha}\,\mathrm{poly}(\lambda)$, where again each element s'_γ in $\mathbf{s}'$ is a block of $\log \lambda$ bits.

Pre-Compute the following for $b = 0$:

$$\mathbf{V}_1 = \left\{\mathsf{Mnml}^{\leq 3}\left(\cup_q \mathsf{Mnml}(s_{q,\gamma})\right)\right\}_{\gamma \in [\Gamma']} \otimes (\mathbf{x}||b||1) \quad (9)$$

$$\mathbf{V}_2 = \left\{\mathsf{Mnml}(s'_\gamma)\right\}_{\gamma \in [\Gamma]} \otimes (b||1) \quad (10)$$

Finally generate:

$$\mathsf{CT} \xleftarrow{\$} \mathsf{FE.Enc}(\mathsf{MPK}, (\mathbf{V}_1, \mathbf{V}_2))$$

KEY GENERATION: $\mathsf{CFE.KeyGen}(\mathsf{MSK}, h)$ does the following:

- Sample $\mathbf{CT} \xleftarrow{\$} \{0,1\}^\ell$, where ℓ is set below.
- Define function g as follows: On input $\mathbf{x}$ of length N, PRG seeds $\mathbf{s}$ and $\mathbf{s}'$ of dimensions described above, and a bit b.

  $g(\mathbf{x}, \mathbf{s}, \mathbf{s}', b)$ does the following:

  - For every $i \in [S]$, let $h_i(\mathbf{x})$ denote the function that computes the i^{th} output bit of $h(\mathbf{x})$. Since $h \in \mathsf{NC}^1_D$, h_i has depth $D(\lambda) = O(\log \lambda)$ and size $2^{D(\lambda)} = \mathrm{poly}(\lambda)$.
  - If $b = 0$, do:
    Expand each row of $\mathbf{s}$ using PRG to obtain a $Q \times S$ matrix $\mathbf{r}$ of pseudo-random bits. That is, let $\mathbf{s}_i$ denote the i^{th} row of $\mathbf{s}$; the i^{th} row $\mathbf{r}_i$ of $\mathbf{r}$ is $\mathrm{PRG}(\mathbf{s}_i)$. Denote by $\mathbf{r}[i]$ the i^{th} column of matrix $\mathbf{r}$, which has length $Q = \mathrm{poly}(\lambda)$.
    For every $i \in [S]$, compute the AIK encoding $\Pi[i]$ of computation $(h_i, \mathbf{x})$ as follows:

    $$\forall\, i \in [S], \qquad \Pi[i] = \mathsf{AIK}(h_i,\ \mathbf{x}\ ;\ \mathbf{r}[i])\ .$$

    Output $\Pi = \{\Pi[i]\}_i$; set $\ell = |\Pi|$.
  - If $b = 1$, output $\Pi = \mathbf{CT} \oplus \mathrm{PRG}(\mathbf{s}')$.
- For every $l \in [\ell]$, let P_l be the degree-L polynomial that on input $(\mathbf{V}_1, \mathbf{V}_2)$ in Equations (9) and (10) computes the l^{th} output bit of $g(\mathbf{x}, \mathbf{s}, \mathbf{s}', b)$.

  For every l, generate a secret key $\mathsf{SK}_l \xleftarrow{\$} \mathsf{FE.KeyGen}(\mathsf{MSK}, P_l)$ for P_l.

Output $\mathsf{SK} = \{\mathsf{SK}_l\}_{l \in [\ell]}$.

DECRYPTION: $\mathsf{CFE.Dec}(\mathsf{SK}, \mathsf{CT})$ computes $\Pi = \{\mathsf{FE.Dec}(\mathsf{SK}_l, \mathsf{CT})\}_{l \in [\ell]}$, parses $\Pi = \{\Pi[i]\}$, and decodes every $\Pi[i]$ using the AIK decoding algorithm to obtain the output $h(\mathbf{x})$.

Figure 3: Single-key Compact FE **CFE** from block-wise locality-L PRG and degree-L FE

Correctness: *For all security parameters $\lambda \in \mathbb{N}$, for every $C \in \mathcal{C}_\lambda$, and every input x, we have that*

$$\Pr[(\sigma, pp) \xleftarrow{\$} \mathsf{SetupIO}(1^\lambda),\ \hat{C} \leftarrow i\mathcal{O}(\mathsf{pp}, C)\ :\ \mathsf{EvalIO}(\sigma, \hat{C}, x) = C(x)] = 1$$

μ-Indistinguishability: *For every ensemble of pairs of circuits $\{C_{0,\lambda}, C_{1,\lambda}\}_{\lambda\in\mathbb{N}}$ satisfying that $C_{b,\lambda} \in \mathcal{C}_\lambda$, $C_{0,\lambda}$ and $C_{1,\lambda}$ have the same size, input-length, and truth table, the following ensembles of distributions are μ-indistinguishable:*

$$\left\{(\sigma, pp) \xleftarrow{\$} \mathsf{SetupIO}(1^\lambda)\ :\ \sigma, \mathsf{pp}, C_{0,\lambda}, C_{1,\lambda}, i\mathcal{O}(\mathsf{pp}, C_{0,\lambda})\right\}_{\lambda\in\mathbb{N}}$$
$$\left\{(\sigma, pp) \xleftarrow{\$} \mathsf{SetupIO}(1^\lambda)\ :\ \sigma, \mathsf{pp}, C_{0,\lambda}, C_{1,\lambda}, i\mathcal{O}(\mathsf{pp}, C_{1,\lambda})\right\}_{\lambda\in\mathbb{N}}$$

Toward proving Proposition 2 for constructing iℓ-bit-input IO for $\mathsf{P/poly}$ in the plain model, we first construct iℓ-bit-input IO for $\mathsf{P/poly}$ in the CRS model as defined below. We say that a polynomial p is upper bounded by another q, if it holds that for all $\lambda \in \mathbb{N}$, $p(\lambda) \le q(\lambda)$.

Definition 28. *Let* iℓ *be any polynomial upper bounded by λ. An* iℓ-bit-input *indistinguishability obfuscator in the CRS model for* $\mathsf{P/poly}$ *is a tuple of uniform* PPT *machines* $(\mathsf{SetupIO}, i\mathcal{O}, \mathsf{EvalIO})$ *satisfying the following condition:* $\mathsf{SetupIO}$ *takes two inputs* $(1^\lambda, 1^l)$*, and for every polynomial l upper bounded by* iℓ*,* $(\mathsf{SetupIO}^l(1^\lambda) = \mathsf{SetupIO}(1^\lambda, 1^{l(\lambda)}), i\mathcal{O}, \mathsf{EvalIO})$ *is an IO scheme in the CRS model for the class of circuits* $\mathcal{C}^l = \{\mathcal{C}^l_\lambda\}_{\lambda\in\mathbb{N}}$ *containing all circuits with size at most λ and input-length at most $l(\lambda)$.*

When i$\ell(\lambda) = \lambda$*, we say that the scheme is a* bounded-input *indistinguishability obfuscator for* $\mathsf{P/poly}$ *in the CRS model.*

It follows from a very simple argument that iℓ-bit-input IO for $\mathsf{P/poly}$ in the CRS model actually implies that in the plain model.

Claim 1. *Let* iℓ *be any polynomial upper bounded by λ. An* iℓ*-bit-input indistinguishability obfuscator for* $\mathsf{P/poly}$ *in the CRS model* $(\mathsf{SetupIO}, i\mathcal{O}, \mathsf{EvalIO})$ *with μ-security, implies an* iℓ*-bit-input indistinguishability obfuscator for* $\mathsf{P/poly}$ *in the plain model $i\mathcal{O}'$ with μ-security.*

Proof. The obfuscator $i\mathcal{O}'$ on input 1^λ and a circuit C with size $|C| < \lambda$ and input-length l works as follows: It samples a fresh reference string and a public parameter $(\sigma, \mathsf{pp}) \xleftarrow{\$} \mathsf{SetupIO}(1^\lambda, 1^l)$ using the input-length of C as the input-length bound, obfuscate C to obtain $\hat{C} \xleftarrow{\$} i\mathcal{O}(\mathsf{pp}, C)$, and outputs $\tilde{C} = (\sigma, \mathsf{pp}, \hat{C})$ as the new obfuscated circuit. The correctness and security of the new obfuscator follows directly from that of the obfuscator in the CRS model. □

4.2.2 IO in the CRS Model from Weakly-Compact FE

We now construct an iℓ-bit-input indistinguishability obfuscator for $\mathsf{P/poly}$ in the CRS model $(\mathsf{SetupIO}, i\mathcal{O}, \mathsf{EvalIO})$, from a public-key weakly-compact FE scheme. To do so, we define the behavior of $\mathsf{SetupIO}$, $i\mathcal{O}$, EvalIO and analyze their correctness, efficiency and security, by induction over the input-length iℓ the algorithms are executed with. More precisely, the construction makes use of the following building blocks:

- Public key FE schemes $\mathbf{CFE} = \{\mathbf{CFE}^{N,D,S}\}$ for $\mathsf{P/poly}$ with $(1-\varepsilon)$-sublinear compact for arbitrary constant $\varepsilon > 0$, where the scheme $\mathbf{CFE}^{N,D,S} = (\mathsf{CFE.Setup}, \mathsf{CFE.KeyGen}, \mathsf{CFE.Enc}, \mathsf{CFE.Dec})$ is a scheme that handles circuits with input-length $N = N(\lambda)$, depth $D = D(\lambda)$, and size $S = S(\lambda)$. Such FE schemes for any polynomials N, D, S are constructed in Section 4.1.

- Two PRGs PRG_1, PRG_2 mapping λ-bit inputs to sufficiently long outputs, of $m_1(\lambda)$-bit and $m_2(\lambda)$-bit respectively. The mapping takes time $\mathrm{poly}(\lambda)m_1(\lambda)$ and $\mathrm{poly}(\lambda)m_2(\lambda)$ respectively. (Such PRGs can be implemented for example using a PRF.)

For any $\lambda \in \mathbb{N}$, we define the behavior of $\mathsf{SetupIO}$, $i\mathcal{O}$, EvalIO when executed with security parameter 1^λ, by induction on the input-length $\mathrm{i}\ell$ they are executed with (more precisely, when $\mathsf{SetupIO}$ is invoked with input $(1^\lambda, 1^{\mathrm{i}\ell})$, and $i\mathcal{O}$ and EvalIO are executed with the produced public parameter and reference string). The scheme is described in Figure 4, which recursively calls itself. Observe that scheme is well-defined, since when it is executed with an input length $\mathrm{i}\ell + 1$, it recursively invokes itself with input length $\mathrm{i}\ell$; in the base case when $\mathrm{i}\ell = 1$, the IO scheme is trivially defined, by letting the obfuscator outputs the only two possible outputs $C(0), C(1)$.

Next, we analyze the correctness, efficiency, and security of the IO scheme in the CRS model defined in Figure 4 by induction, in Lemma 6, 7, and 8.

Lemma 6 (Induction on Correctness). *For any $\lambda \in \mathbb{N}$ and any $\mathrm{i}\ell \in \mathbb{N}$, if the IO scheme $(\mathsf{SetupIO}, i\mathcal{O}, \mathsf{EvalIO})$ in Figure 4 satisfy correctness when executed with security parameter 1^λ and input-length $1^{\mathrm{i}\ell}$, for any circuit C with size at most λ and input-length $\mathrm{i}\ell$, then the scheme also satisfies correctness when executed with 1^λ and $1^{\mathrm{i}\ell+1}$, for any circuit C with size at most λ and input-length $\mathrm{i}\ell + 1$.*

Proof. When executed with $1^\lambda, 1^{\mathrm{i}\ell+1}$, the setup algorithm $\mathsf{SetupIO}$ recursively calls itself with input $(1^\lambda, 1^{\mathrm{i}\ell})$ to obtain $(\sigma^{\mathrm{i}\ell}, \mathsf{pp}^{\mathrm{i}\ell})$. Moreover, it also calls the setup algorithm of the FE scheme to obtain $(\mathsf{MPK}, \mathsf{MSK})$, and generates a secret key SK for a function f, which on input $(C, \mathrm{s}_0, \mathrm{s}_1, 0^\lambda, 0)$ obfuscates the circuit C with the first input-bit fixed to 0 and 1, namely $C^{(0)} = C(0, \star)$ and $C^{(1)} = C(1, \star)$, using $\mathsf{pp}^{\mathrm{i}\ell}$ (and pseudorandomness generated by a PRG). Finally, the reference string and public parameter are set to $\sigma^{\mathrm{i}\ell+1} = (\sigma^{\mathrm{i}\ell}, \mathsf{SK})$ and $\mathsf{pp}^{\mathrm{i}\ell+1} = (\mathsf{pp}^{\mathrm{i}\ell}, \mathsf{MPK})$.

The obfuscation of a circuit C with size at most λ and input-length $\mathrm{i}\ell+1$ generated with public parameter $\mathsf{pp}^{\mathrm{i}\ell+1}$ is simply a FE ciphertext CT of $(C, \mathrm{s}_0, \mathrm{s}_1, 0^\lambda, 0)$ with fresh random PRG seeds.

Evaluation decrypts the ciphertext CT using SK in $\sigma^{\mathrm{i}\ell+1}$. By the correctness of the FE scheme, decryption outputs $f(C, \mathrm{s}_0, \mathrm{s}_1, 0^\lambda, 0) = (\hat{C}^{(0)}, \hat{C}^{(1)})$, where $\hat{C}^{(d)}$ is an obfuscation of $C^{(d)}$ generated using $\mathsf{pp}^{\mathrm{i}\ell}$. Thus, by the correctness of the IO scheme when executed with input length $\mathrm{i}\ell$, for any $x \in \{0,1\}^{\mathrm{i}\ell+1}$, evaluation outputs the right output $C_{x_1}(x_{\geq 2}) = C(x)$, which concludes the proof of the lemma. ☐

Lemma 7 (Induction on Efficiency). *There exists sufficiently large universal polynomials P, Q, and T, such that, the following holds for the IO scheme $(\mathsf{SetupIO}, i\mathcal{O}, \mathsf{EvalIO})$ in Figure 4: For every λ and input-length $\mathrm{i}\ell$,*

- *Suppose that the scheme when executed with $1^\lambda, 1^{\mathrm{i}\ell}$ for* arbitrary *circuit C with size at most λ and input-length $\mathrm{i}\ell$, and input $x \in \{0,1\}^{\mathrm{i}\ell}$, satisfy the following efficiency*

$$
\begin{aligned}
(\sigma^{\mathrm{i}\ell}, \mathsf{pp}^{\mathrm{i}\ell}) &\xleftarrow{\$} \mathsf{SetupIO}(1^\lambda, 1^{\mathrm{i}\ell}) && \mathsf{Time}_{\mathsf{SetupIO}}(1^\lambda, 1^{\mathrm{i}\ell}) \leq \mathrm{i}\ell \times Q(\lambda)\ , \\
\hat{C} &\xleftarrow{\$} i\mathcal{O}(\mathsf{pp}^{\mathrm{i}\ell}, C) && \mathsf{Time}_{i\mathcal{O}}(\mathsf{pp}^{\mathrm{i}\ell}, C) \leq P(\lambda)\ , \\
y &= \mathsf{EvalIO}(\sigma^{\mathrm{i}\ell}, \hat{C}, x) && \mathsf{Time}_{\mathsf{EvalIO}}(\sigma^{\mathrm{i}\ell}, \hat{C}, x) \leq \mathrm{i}\ell \times T(\lambda)\ .
\end{aligned}
$$

- *Then, they satisfy the following efficiency when executed with $1^\lambda, 1^{\mathrm{i}\ell+1}$ for arbitrary circuit C with size at most λ and input-length $\mathrm{i}\ell + 1$, and input $x \in \{0,1\}^{\mathrm{i}\ell+1}$,*

$$
\begin{aligned}
(\sigma^{\mathrm{i}\ell+1}, \mathsf{pp}^{\mathrm{i}\ell+1}) &\xleftarrow{\$} \mathsf{SetupIO}(1^\lambda, 1^{\mathrm{i}\ell+1}) && \mathsf{Time}_{\mathsf{SetupIO}}(1^\lambda, 1^{\mathrm{i}\ell+1}) \leq (\mathrm{i}\ell+1) \times Q(\lambda)\ , \\
\hat{C} &\xleftarrow{\$} i\mathcal{O}(\mathsf{pp}^{\mathrm{i}\ell+1}, C) && \mathsf{Time}_{i\mathcal{O}}(\mathsf{pp}^{\mathrm{i}\ell+1}, C) \leq P(\lambda)\ , \\
y &= \mathsf{EvalIO}(\sigma^{\mathrm{i}\ell+1}, \hat{C}, x) && \mathsf{Time}_{\mathsf{EvalIO}}(\sigma^{\mathrm{i}\ell+1}, \hat{C}, x) \leq (\mathrm{i}\ell+1) \times T(\lambda)\ .
\end{aligned}
$$

An IO scheme $(\mathsf{SetupIO}, i\mathcal{O}, \mathsf{EvalIO})$ **in the CRS model**

Base Case — Input-length $\mathrm{i}\ell = 1$

- $\mathsf{SetupIO}(1^\lambda, 1)$ outputs $(\sigma^1, \mathsf{pp}^1) = (\mathsf{base}, \mathsf{base})$.
- $i\mathcal{O}(\mathsf{pp}^1 = \mathsf{base}, C)$ outputs $(C(0), C(1))$.
- $i\mathcal{O}(\sigma^1 = \mathsf{base}, (y_0, y_1), x)$ outputs $\perp$ if x is not a single bit, and otherwise outputs y_x.

Induction — From Input-length $\mathrm{i}\ell$ **to** $\mathrm{i}\ell + 1$

- $\mathsf{SetupIO}(1^\lambda, 1^{\mathrm{i}\ell+1})$ does:
  - Recursively setup the scheme for input-length $\mathrm{i}\ell$ to obtain $(\sigma^{\mathrm{i}\ell}, \mathsf{pp}^{\mathrm{i}\ell}) \xleftarrow{\$} \mathsf{SetupIO}(1^\lambda, 1^{\mathrm{i}\ell})$.
  - Sample a random string $\mathbf{CT} \xleftarrow{\$} \{0,1\}^\ell$ of length ℓ set below.
  - Sample master keys of the weakly-compact FE scheme $(\mathsf{MPK}, \mathsf{MSK}) \xleftarrow{\$} \mathsf{CFE.Setup}(1^\lambda)$.
  - Generate a FE secret key $\mathsf{SK}_f \xleftarrow{\$} \mathsf{CFE.KeyGen}(\mathsf{MSK}, f)$ for function f defined as follows:

    $f(C, \mathbf{s}_0, \mathbf{s}_1, \mathbf{s}', b)$ does the following, on input a circuit C of size at most S and input length at most $l + 1$, PRG seeds $\mathbf{s}_0, \mathbf{s}_1, \mathbf{s}'$ of length λ, and a bit b.
    - $*$ If $b = 0$, let $C^{(d)} = C(d, \star)$ denote the circuit C with the first input bit fixed to d Recursively call the IO scheme to obfuscate $C^{(0)}$ and $C^{(1)}$, which have ℓ-bit input, using pseudo-randomness generated through a PRG,

      $$\left\{\hat{C}^{(d)} = i\mathcal{O}(\mathsf{pp}^{\mathrm{i}\ell}, C^{(d)}\ ;\ \mathrm{PRG}_1(\mathbf{s}_d))\right\}_{d \in \{0,1\}} .$$

      Output $z = (\hat{C}^{(0)}, \hat{C}^{(1)})$; set $\ell = |z|$.
    - $*$ If $b = 1$, output $z = \mathbf{CT} \oplus \mathrm{PRG}_2(\mathbf{s}')$.
  - Output $\sigma^{\mathrm{i}\ell+1} = (\sigma^{\mathrm{i}\ell}, \mathsf{SK}_f)$ and $\mathsf{pp}^{\mathrm{i}\ell+1} = (\mathsf{pp}^{\mathrm{i}\ell}, \mathsf{MPK})$.

- $i\mathcal{O}(\mathsf{pp}^{\mathrm{i}\ell+1} = (\mathsf{pp}^{\mathrm{i}\ell}, \mathsf{MPK}), C)$ does:

  Sample two PRG seeds $\mathbf{s}_0, \mathbf{s}_1 \xleftarrow{\$} \{0,1\}^\lambda$, and encrypt $(C, \mathbf{s}_0, \mathbf{s}_1, 0^\lambda, 0)$ using FE,

  $$\mathsf{CT} \xleftarrow{\$} \mathsf{CFE.Enc}(\mathsf{MPK}, (C, \mathbf{s}_0, \mathbf{s}_1, 0^\lambda, 0))$$

  Output the ciphertext $\hat{C} = \mathsf{CT}$ as the obfuscated circuit.

- $\mathsf{EvalIO}(\sigma^{\mathrm{i}\ell+1} = (\sigma^{\mathrm{i}\ell}, \mathsf{SK}), \hat{C} = \mathsf{CT}, x)$ does:
  - Decrypt the FE ciphertext and secret key $z = \mathsf{CFE.Dec}(\mathsf{SK}, \mathsf{CT})$; parse z as $(\hat{C}^{(0)}, \hat{C}^{(1)})$.
  - Recursively call the IO scheme to evaluate $y = \mathsf{EvalIO}(\sigma^{\mathrm{i}\ell}, \hat{C}_{x_1}, x_{\geq 2})$, where $x_{\geq 2}$ are the last l bits of x.

Figure 4: An IO scheme $(\mathsf{SetupIO}, i\mathcal{O}, \mathsf{EvalIO})$ in the CRS model from weakly-compact public-key FE.

Proof. To analyze the efficiency of the IO scheme in an execution with $1^\lambda, 1^{i\ell+1}$, circuit C, and input x as specified in the lemma statement, we start with analyzing the size S and input-length N of the circuit that computes the function f for which the FE secret key is generated. The function f takes input of form $(\bar{C}, \mathbf{s}_0, \mathbf{s}_1, \mathbf{s}', b)$ for any circuit with size $|\bar{C}| \leq \lambda$ and input-length $i\ell + 1$, and PRG seeds of length λ. Thus, its input-length is

$$N = N(\lambda) \quad = \quad 4\lambda + 1 \; .$$

If the input bit $b = 0$, f obfuscates $\bar{C}^{(0)}$ and $\bar{C}^{(1)}$ using the IO scheme with a public parameter $\mathsf{pp}^{i\ell}$ generated for input-length $i\ell$ and pesudorandomness produced by $\mathrm{PRG}_1(\mathbf{s}_0), \mathrm{PRG}_1(\mathbf{s}_1)$. By our hypothesis on the efficiency of the IO scheme for input-length $i\ell$, and the efficiency of PRG_1, this step takes at most time $\mathrm{poly}(\lambda) \times P(\lambda)$. If $b = 1$, f "decrypts" a hardwired one-time-pad ciphertext $\mathbf{CT}$ using pseudorandom pad generated by $\mathrm{PRG}_2(\mathbf{s}')$. The length of the ciphertext $\mathbf{CT}$ is the same as the length of the obfuscation of $\bar{C}^{(0)}$ and $\bar{C}^{(1)}$, and hence is bounded by $2P(\lambda)$. Thus, by the efficiency of PRG_2, this step takes at most time $\mathrm{poly}(\lambda) \times P(\lambda)$. In summary, f can be computed by a circuit with size

$$S \leq \mathrm{poly}(\lambda) \times P(\lambda) \; .$$

We now analyze the runtime of SetupIO, $i\mathcal{O}$, and EvalIO, when executed with $1^\lambda, 1^{i\ell+1}$, C with input-length $i\ell + 1$, and $x \in \{0,1\}^{i\ell+1}$.

$$\begin{aligned}
\mathsf{Time}_{\mathsf{SetupIO}}(1^\lambda, 1^{i\ell+1}) \quad &= \quad \mathsf{Time}_{\mathsf{SetupIO}}(1^\lambda, 1^{i\ell}) + \mathsf{Time}_{\mathsf{CFE.Setup}}(1^\lambda) + \mathsf{Time}_{\mathsf{CFE.KeyGen}}(\mathsf{MSK}, f) + \mathrm{poly}(\lambda) \\
&= \quad i\ell \times Q(\lambda) + \mathrm{poly}(\lambda) + \mathrm{poly}(\lambda, S) + \mathrm{poly}(\lambda) \\
&\leq \quad i\ell \times Q(\lambda) + \mathrm{poly}(\lambda, \mathrm{poly}(\lambda)P(\lambda)) + \mathrm{poly}(\lambda) \\
&\leq \quad (i\ell + 1) \times Q(\lambda)
\end{aligned}$$

where the second equality follows form the hypothesis on the runtime of SetupIO with input-length $i\ell$ and the efficiency of the FE scheme, the first inequality follows from plugging in the bound on S, and the last inequality follows if $Q(\lambda)$ is sufficiently large (comparing to P).

$$\begin{aligned}
\mathsf{Time}_{i\mathcal{O}}(\mathsf{pp}^{i\ell+1}, C) \quad &= \quad \mathsf{Time}_{\mathsf{CFE.Enc}}(\mathsf{MPK}, (C, \mathbf{s}_0, \mathbf{s}_1, 0^\lambda, 0)) + \mathrm{poly}(\lambda) \\
&= \quad \mathrm{poly}(\lambda, N) S^{1-\varepsilon} + \mathrm{poly}(\lambda) \\
&\leq \quad \mathrm{poly}(\lambda)(\mathrm{poly}(\lambda)P(\lambda))^{1-\varepsilon} + \mathrm{poly}(\lambda) \\
&\leq \quad P(\lambda)
\end{aligned}$$

where the second equality follows from the $(1-\varepsilon)$-sublinear compactness of the FE scheme, the first inequality follows from plugging the bound on S, and the last inequality holds when P is sufficiently large.

$$\begin{aligned}
\mathsf{Time}_{\mathsf{EvalIO}}(\sigma^{i\ell+1}, \hat{C}, x) \quad &= \quad \mathsf{Time}_{\mathsf{CFE.Dec}}(\mathsf{SK}, \mathsf{CT}) + \mathsf{Time}_{\mathsf{EvalIO}}(\sigma^{i\ell}, \hat{C}_{x_1}, x_{\geq 2}) + \mathrm{poly}(\lambda) \\
&= \quad \mathrm{poly}(\lambda, S) + i\ell \times T(\lambda) + \mathrm{poly}(\lambda) \\
&\leq \quad \mathrm{poly}(\lambda, \mathrm{poly}(\lambda)P(\lambda)) + i\ell \times T(\lambda) + \mathrm{poly}(\lambda) \\
&\leq \quad (i\ell + 1) \times T(\lambda)
\end{aligned}$$

where the second equality follows from the efficiency of FE decryption, and the hypothesis on the runtime of EvalIO for evaluating obfuscated circuits with $i\ell$-bit inputs, the first inequality follows from plugging in the bound on S, and last inequality holds when T is sufficiently large (comparing to P).

We conclude the proof. $\square$

Lemma 8 (Induction on Security). *Let $\mu, \mu_{\mathrm{FE}}, \mu_{\mathrm{PRG}}$ be any functions from $\mathbb{N}$ to $[0,1]$. Suppose that the following holds:*

- *The FE scheme* $\mathbf{CFE}$ *is* μ_{FE}-$\mathsf{Full\text{-}Sel}$-*secure.*
- *The PRGs* PRG_1 *and* PRG_2 *are both* μ_{PRG}*-indistinguishable.*
- *For every efficient distinguisher D, every $\lambda \in \mathbb{N}$, $\mathrm{i}\ell < \lambda \in \mathbb{N}$, C_0, C_1 with size $|C_0| = |C_1| \le \lambda$, input-length $\mathrm{i}\ell$, and identical truth table, the following holds.*

$$\begin{aligned} &\Big| \Pr[(\sigma^{\mathrm{i}\ell}, pp^{\mathrm{i}\ell}) \xleftarrow{\$} \mathsf{SetupIO}(1^\lambda, 1^{\mathrm{i}\ell}) \ : \ D(\sigma^{\mathrm{i}\ell}, \mathsf{pp}^{\mathrm{i}\ell}, C_0, C_1, i\mathcal{O}(\mathsf{pp}^{\mathrm{i}\ell}, C_0)) = 1] \\ = \quad &\Pr[(\sigma^{\mathrm{i}\ell}, pp^{\mathrm{i}\ell}) \xleftarrow{\$} \mathsf{SetupIO}(1^\lambda, 1^{\mathrm{i}\ell}) \ : \ D(\sigma^{\mathrm{i}\ell}, \mathsf{pp}^{\mathrm{i}\ell}, C_0, C_1, i\mathcal{O}(\mathsf{pp}^{\mathrm{i}\ell}, C_1)) = 1] \Big| \ \le \ \mu(\lambda) \end{aligned}$$

Then, for every efficient distinguisher D, $\lambda \in \mathbb{N}$, $\mathrm{i}\ell + 1 < \lambda \in \mathbb{N}$, C_0, C_1 with size $|C_0| = |C_1| \le \lambda$, input-length $\mathrm{i}\ell + 1$, and identical truth table, the following holds.

$$\begin{aligned} &\Big| \Pr[(\sigma^{\mathrm{i}\ell+1}, pp^{\mathrm{i}\ell+1}) \xleftarrow{\$} \mathsf{SetupIO}(1^\lambda, 1^{\mathrm{i}\ell+1}) \ : \ D(\sigma^{\mathrm{i}\ell+1}, \mathsf{pp}^{\mathrm{i}\ell+1}, C_0, C_1, i\mathcal{O}(\mathsf{pp}^{\mathrm{i}\ell+1}, C_0)) = 1] \\ - \quad &\Pr[(\sigma^{\mathrm{i}\ell+1}, pp^{\mathrm{i}\ell+1}) \xleftarrow{\$} \mathsf{SetupIO}(1^\lambda, 1^{\mathrm{i}\ell+1}) \ : \ D(\sigma^{\mathrm{i}\ell+1}, \mathsf{pp}^{\mathrm{i}\ell+1}, C_0, C_1, i\mathcal{O}(\mathsf{pp}^{\mathrm{i}\ell+1}, C_1)) = 1] \Big| \\ \le \quad & 2\mu(\lambda) + 2\mu_{FE}(\lambda) + 6\mu_{\mathrm{PRG}}(\lambda) \end{aligned}$$

Proof. Fix any efficient distinguisher D, λ, $\mathrm{i}\ell+1 < \lambda$, C_0, C_1 with size $|C_0| = |C_1| \le \lambda$, input-length $\mathrm{i}\ell + 1$, and identical truth table, we prove the lemma through a sequence of hybrids.

Hybrid Distribution $\mathcal{D}_0^b$: Sample $(\sigma^{\mathrm{i}\ell+1}\mathsf{pp}^{\mathrm{i}\ell+1}, \hat{C})$ using the IO scheme honestly, obfuscating the circuit C_b, that is,

$$\mathcal{D}_0^b \ : \ (\sigma^{\mathrm{i}\ell+1}, \mathsf{pp}^{\mathrm{i}\ell+1}) \xleftarrow{\$} \mathsf{SetupIO}(1^\lambda, 1^{\mathrm{i}\ell+1}); \ \hat{C} \xleftarrow{\$} i\mathcal{O}(\mathsf{pp}^{\mathrm{i}\ell+1}, C_b) \ .$$

Parse $\sigma^{\mathrm{i}\ell+1} = (\sigma^{\mathrm{i}\ell}, \mathsf{SK})$, $\mathsf{pp}^{\mathrm{i}\ell+1} = (\mathsf{pp}^{\mathrm{i}\ell}, \mathsf{MPK})$, and $\hat{C} = \mathsf{CT}$.

Hybrid Distribution $\mathcal{D}_1^b$: Sample $(\sigma^{\mathrm{i}\ell+1}\mathsf{pp}^{\mathrm{i}\ell+1}, \hat{C})$ identically to $\mathcal{D}_0^b$, except that the secret key SK in $\sigma^{\mathrm{i}\ell+1}$ is generated differently.

In $\mathcal{D}_0^b$, the SK is associated with a function f that is hardwired with a random string $\mathbf{CT}$, and the obfuscated circuit $\hat{C}$ is a ciphertext CT encrypting input $(C_b, \mathrm{s}_0, \mathrm{s}_1, 0^\lambda, 0)$. During evaluation, CT is decrypted by SK producing $f(C_b, \mathrm{s}_0, \mathrm{s}_1, 0^\lambda, 0)$, which by definition of f is

$$f(C_b, \mathrm{s}_0, \mathrm{s}_1, 0^\lambda, 0) = \Big(\hat{C}^{(d)} = i\mathcal{O}(\mathsf{pp}^{\mathrm{i}\ell}, C_b^{(d)} \ ; \ \mathrm{PRG}_1(\mathrm{s}_d)) \Big)_{d \in \{0,1\}} \ . \tag{11}$$

In $\mathcal{D}_1^b$, $\mathbf{CT}$ is no longer a random string, and instead is set to a one-time-pad ciphertext encrypting $(\hat{C}^{(0)}, \hat{C}^{(1)})$ as described above using pad $\mathrm{PRG}_2(\mathrm{s}')$, that is,

$$\mathbf{CT} = (\hat{C}^{(0)}, \hat{C}^{(1)}) \oplus \mathrm{PRG}_2(\mathrm{s}') \ .$$

The only difference between $\mathcal{D}_0^b$ and $\mathcal{D}_1^b$ is whether $\mathbf{CT}$ is a random string or a one-time-pad ciphertext with pad $\mathrm{PRG}_2(\mathbf{s}')$. Since the seed $\mathbf{s}'$ is not used anywhere else in $\mathcal{D}_0^b$ and $\mathcal{D}_1^b$, by the μ_{PRG}-indistinguishability of PRG_2, we have that the distinguisher D distinguishes these two distributions with advantage at most $\mu_{\mathrm{PRG}}(\lambda)$, that is,

$$\begin{aligned} & \Big| \Pr[(\sigma^{\mathrm{i}\ell+1}, pp^{\mathrm{i}\ell+1}, \hat{C}) \xleftarrow{\$} \mathcal{D}_0^b \ : \ D(\sigma^{\mathrm{i}\ell+1}, \mathsf{pp}^{\mathrm{i}\ell+1}, C_0, C_1, \hat{C}) = 1] \\ - \quad & \Pr[(\sigma^{\mathrm{i}\ell+1}, pp^{\mathrm{i}\ell+1}, \hat{C}) \xleftarrow{\$} \mathcal{D}_1^b \ : \ D(\sigma^{\mathrm{i}\ell+1}, \mathsf{pp}^{\mathrm{i}\ell+1}, C_0, C_1, \hat{C}) = 1] \Big| \ \le \ \mu_{\mathrm{PRG}}(\lambda) \ . \end{aligned}$$

Hybrid $\mathcal{D}_2^b$: Sample $(\sigma^{\mathrm{i}\ell+1}\mathsf{pp}^{\mathrm{i}\ell+1}, \hat{C})$ identically to $\mathcal{D}_1^b$, except that the obfuscated circuit $\hat{C}$ is generated differently.

In $\mathcal{D}_1^b$, $\hat{C}$ is a FE ciphertext CT encrypting $(C_b, \mathbf{s}_0, \mathbf{s}_1, 0^\lambda, 0)$,

$$\hat{C} = \mathsf{CT} \xleftarrow{\$} \mathsf{CFE.Enc}(\mathsf{MPK}, (C_b, \mathbf{s}_0, \mathbf{s}_1, 0^\lambda, 0)) \ .$$

In $\mathcal{D}_2^b$, $\hat{C}$ becomes a ciphertext CT of $(C_b, 0^\lambda, 0^\lambda, \mathbf{s}', 1)$,

$$\hat{C} = \mathsf{CT} \xleftarrow{\$} \mathsf{CFE.Enc}(\mathsf{MPK}, (C_b, 0^\lambda, 0^\lambda, \mathbf{s}', 1)) \ .$$

Note that in both $\mathcal{D}_1^b$ and $\mathcal{D}_2^b$, $\mathbf{CT}$ is set to $(\hat{C}^{(0)}, \hat{C}^{(1)}) \oplus \mathrm{PRG}_2(\mathbf{s}')$. With such $\mathbf{CT}$, by definition of f,

$$f(C_b, \mathbf{s}_0, \mathbf{s}_1, 0^\lambda, 0) = f(C_b, 0^\lambda, 0^\lambda, \mathbf{s}', 1) \ .$$

Therefore, by the μ_{FE}-Full-Sel-security of $\mathbf{CFE}$, we have that the distinguisher D distinguishes distributions $\mathcal{D}_1^b$ and $\mathcal{D}_2^b$ with advantage at most $\mu_{\mathrm{FE}}(\lambda)$, that is,

$$\begin{aligned} & \Big| \Pr[(\sigma^{\mathrm{i}\ell+1}, pp^{\mathrm{i}\ell+1}, \hat{C}) \xleftarrow{\$} \mathcal{D}_1^b \ : \ D(\sigma^{\mathrm{i}\ell+1}, \mathsf{pp}^{\mathrm{i}\ell+1}, C_0, C_1, \hat{C}) = 1] \\ - \quad & \Pr[(\sigma^{\mathrm{i}\ell+1}, pp^{\mathrm{i}\ell+1}, \hat{C}) \xleftarrow{\$} \mathcal{D}_2^b \ : \ D(\sigma^{\mathrm{i}\ell+1}, \mathsf{pp}^{\mathrm{i}\ell+1}, C_0, C_1, \hat{C}) = 1] \Big| \ \le \ \mu_{\mathrm{FE}}(\lambda) \ . \end{aligned}$$

Hybrid $\mathcal{D}_3^b$: Sample $(\sigma^{\mathrm{i}\ell+1}\mathsf{pp}^{\mathrm{i}\ell+1}, \hat{C})$ identically to $\mathcal{D}_2^b$, except that again the string $\mathbf{CT}$ hardwired in the function f associated with the FE secret key SK in $\sigma^{\mathrm{i}\ell+1}$ is generated differently.

In $\mathcal{D}_2^b$, $\mathbf{CT}$ is the one-time-pad encryption of the obfuscation $(\hat{C}^{(0)}, \hat{C}^{(1)})$ of $(C_b^{(0)}, C_b^{(1)})$ using pseudorandomness generated by $\mathrm{PRG}_1(\mathbf{s}_0)$ and $\mathrm{PRG}_1(\mathbf{s}_1)$, as described in Equation (11).

In $\mathcal{D}_3^b$, $\mathbf{CT}$ is the one-time-pad encryption of still obfuscation $(\tilde{C}^{(0)}, \tilde{C}^{(1)})$ of $(C_b^{(0)}, C_b^{(1)})$, but generated with truly random coins $\mathbf{r}_0, \mathbf{r}_1$,

$$\Big(\tilde{C}^{(d)} = i\mathcal{O}(\mathsf{pp}^{\mathrm{i}\ell}, C_b^{(d)} \ ; \ \mathbf{r}_d) \Big)_{d \in \{0,1\}} \ , \qquad \mathbf{CT} = (\tilde{C}^{(0)}, \tilde{C}^{(1)}) \oplus \mathrm{PRG}_2(\mathbf{s}') \ .$$

The only difference between $\mathcal{D}_2^b$ and $\mathcal{D}_3^b$ is whether the encrypted obfuscation are generated using pseudorandomness $\mathrm{PRG}_1(\mathbf{s}_0), \mathrm{PRG}_1(\mathbf{s}_1)$ or using true randomness $\mathbf{r}_0, \mathbf{r}_1$. Since in both distributions $\mathbf{s}_0, \mathbf{s}_1$ are not used anywhere else, it follows from the μ_{PRG}-indistinguishability of the PRGs that

$$\begin{aligned} & \Big| \Pr[(\sigma^{\mathrm{i}\ell+1}, pp^{\mathrm{i}\ell+1}, \hat{C}) \xleftarrow{\$} \mathcal{D}_2^b \ : \ D(\sigma^{\mathrm{i}\ell+1}, \mathsf{pp}^{\mathrm{i}\ell+1}, C_0, C_1, \hat{C}) = 1] \\ - \quad & \Pr[(\sigma^{\mathrm{i}\ell+1}, pp^{\mathrm{i}\ell+1}, \hat{C}) \xleftarrow{\$} \mathcal{D}_3^b \ : \ D(\sigma^{\mathrm{i}\ell+1}, \mathsf{pp}^{\mathrm{i}\ell+1}, C_0, C_1, \hat{C}) = 1] \Big| \ \le \ 2\mu_{\mathrm{PRG}}(\lambda) \ . \end{aligned}$$

Next, observe that the only difference between $\mathcal{D}_3^0$ and $\mathcal{D}_3^1$ lies in that in the former, $(\tilde{C}^{(0)}, \tilde{C}^{(1)})$ obfuscate $(C_0^{(0)}, C_0^{(1)})$, whereas in the latter they obfuscate $(C_1^{(0)}, C_1^{(1)})$. By the fact that C_0 and C_1 are equivalent (in the sense of having the same size, input-length, and truth table), so are $C_0^{(d)}$ and $C_1^{(d)}$ for any $d \in \{0,1\}$. Therefore, for any d, by our hypothesis on the security of the IO scheme for circuits with iℓ-bit inputs, the obfuscation of $C_0^{(d)}$ and $C_1^{(d)}$ with input-length iℓ is $\mu(\lambda)$ indistinguishable. Hence,

$$\begin{aligned} & \Big| \Pr[(\sigma^{\mathrm{i}\ell+1}, pp^{\mathrm{i}\ell+1}, \hat{C}) \xleftarrow{\$} \mathcal{D}_3^0 \ : \ D(\sigma^{\mathrm{i}\ell+1}, \mathsf{pp}^{\mathrm{i}\ell+1}, C_0, C_1, \hat{C}) = 1] \\ - \ & \Pr[(\sigma^{\mathrm{i}\ell+1}, pp^{\mathrm{i}\ell+1}, \hat{C}) \xleftarrow{\$} \mathcal{D}_3^1 \ : \ D(\sigma^{\mathrm{i}\ell+1}, \mathsf{pp}^{\mathrm{i}\ell+1}, C_0, C_1, \hat{C}) = 1] \Big| \ \leq \ 2\mu(\lambda) \end{aligned}$$

Finally, it follows from a hybrid argument that

$$\begin{aligned} & \Big| \Pr[(\sigma^{\mathrm{i}\ell+1}, pp^{\mathrm{i}\ell+1}, \hat{C}) \xleftarrow{\$} \mathcal{D}_0^0 \ : \ D(\sigma^{\mathrm{i}\ell+1}, \mathsf{pp}^{\mathrm{i}\ell+1}, C_0, C_1, \hat{C}) = 1] \\ - \ & \Pr[(\sigma^{\mathrm{i}\ell+1}, pp^{\mathrm{i}\ell+1}, \hat{C}) \xleftarrow{\$} \mathcal{D}_0^1 \ : \ D(\sigma^{\mathrm{i}\ell+1}, \mathsf{pp}^{\mathrm{i}\ell+1}, C_0, C_1, \hat{C}) = 1] \Big| \ \leq \ 2\mu(\lambda) + 2\mu_{\mathrm{FE}}(\lambda) + 6\mu_{\mathrm{PRG}}(\lambda) \end{aligned}$$

This concludes the proof of the lemma. □

With Lemma 6, 7, and 8, we now prove Proposition 2.

Proof of Proposition 2. Assume the existence of 1-key weakly-compact public-key FE for $\mathsf{P/poly}$, with $2^{-\mathrm{i}\ell(\lambda)+\kappa(\lambda)}\,\mathrm{negl}(\lambda)$-security. This implies the existence of PRGs with $2^{-\mathrm{i}\ell(\lambda)+\kappa(\lambda)}\,\mathrm{negl}(\lambda)$-indistinguishability. That is,

$$\mu_{\mathrm{FE}}(\lambda) = \mu_{\mathrm{PRG}}(\lambda) = \delta = 2^{-\mathrm{i}\ell(\lambda)+\kappa(\lambda)}\,\mathrm{negl}(\lambda)$$

To show that there exists i$\ell(\lambda)$-bit-input IO for $\mathsf{P/poly}$, with $2^{-\kappa(\lambda)}\,\mathrm{negl}(\lambda)$-security, by Claim 1, it suffices to show that there exists i$\ell(\lambda)$-bit-input IO for $\mathsf{P/poly}$ in the CRS model with $2^{-\kappa(\lambda)}\,\mathrm{negl}(\lambda)$-security. We argue that the construction in Figure 4 is such a scheme.

First observe that in the base case, when $(\mathsf{SetupIO}, i\mathcal{O}, \mathsf{EvalIO})$ are executed with $(1^\lambda, 1)$, the scheme is correct and efficient (for any circuit with size $\leq \lambda$ and single-bit inputs and any input bit x), and perfectly secure with distinguishing gap 0 (for any two single-bit-input circuits with the same size $\leq \lambda$ and truth tables). In other words, when i$\ell = 1$, the scheme satisfy the premises of Lemma 6, 7, and 8.

Therefore, applying these lemmas, we have that for every λ and every $l \leq \mathrm{i}\ell(\lambda)$, when $(\mathsf{SetupIO}, i\mathcal{O}, \mathsf{EvalIO})$ are executed with $(1^\lambda, 1^l)$, the scheme is still correct, efficient, and has distinguishing gap μ_l bounded as below.

$$\begin{aligned} \mu_l \ &\leq \ 2\mu_{l-1} + 2\mu_{\mathrm{FE}} + 6\mu_{\mathrm{PRG}} = 2\mu_{l-1} + 8\delta \\ &\leq \ 2(2\mu_{l-2} + 8\delta) + 8\delta \ \leq \ \cdots \ \leq \ 2^{l-1}\mu_1 + \left(\sum_{i=0}^{l-2} 2^i\right) 8\delta \\ &\leq \ 2^{l+2}\delta = 2^{l+2}2^{-\mathrm{i}\ell+\kappa}\,\mathrm{negl}(\lambda) = 2^{-(\mathrm{i}\ell-l+\kappa)}\,\mathrm{negl}(\lambda) \end{aligned}$$

The fifth inequality follows from the fact that $\mu_1 = 0$. Thus, $\mu_{\mathrm{i}\ell} = 2^{-\kappa}\,\mathrm{negl}(\lambda)$, which concludes the proposition. □

5 FE from $\omega(\log \lambda)$-Bit-Input IO for $\mathsf{P/poly}$

In this section, we show Theorem 4, i.e., we prove via a new transformation that adaptively-secure collusion-resistant public-key functional encryption for $\mathsf{P/poly}$ is implied by IO for circuits with short, $\omega(\log \lambda)$-bit, inputs and public key encryption, both with slightly super-polynomial security. Note that, in contrast, previous constructions of collusion-resistant FE for $\mathsf{P/poly}$ either rely on multilinear maps [GGHZ16], or require IO for all $\mathsf{P/poly}$, including circuits with long (polynomial) inputs [GGH+13b].

Our proof generically transforms any *1-key* (public key) FE scheme for any circuit class $\mathcal{C}$ into a *collusion-resistant* (public key) FE scheme for the same circuit class, using IO for circuits with $\omega(\lambda)$-bit inputs. The encryption time of the resulting FE schemes is polynomial in the encryption time of the original schemes, and hence if the original scheme is (non-)compact, so is the resulting FE scheme. The transformation also preserves the same type of security — namely $\mathsf{Full\text{-}Sel}$- or Adap-security— but incurs a $2^{\omega(\lambda)}$ security loss.

More precisely, we prove the following below in Section 5.1.

Proposition 3. *Let $\mathcal{C}$ be any circuit class, τ be any polynomial, and $\mathrm{i}\ell$ be any polynomial such that $\mathrm{i}\ell(\lambda) = \omega(\log \lambda) \leq \lambda$. Assume the existence of an $\mathrm{i}\ell(\lambda)$-bit-input indistinguishability obfuscator $i\mathcal{O}$ for* $\mathsf{P/poly}$. *Then, any* 1-key *public-key FE schemes* $\mathbf{OFE}$ *for $\mathcal{C}$ can be generically transformed into* collusion-resistant *FE schemes* $\mathbf{CRFE}$ *for $\mathcal{C}$, with the following properties:*

- *The encryption time of* $\mathbf{CRFE}$ *is polynomial in the encryption time of* $\mathbf{OFE}$.
- *If $i\mathcal{O}$ is $2^{-(\mathrm{i}\ell(\lambda)+\tau(\lambda))}\,\mathrm{negl}(\lambda)$-secure and* $\mathbf{OFE}$ *is $2^{-(\mathrm{i}\ell(\lambda)+\tau(\lambda))}\,\mathrm{negl}(\lambda)$-(*$\mathsf{Adap}$ *or* $\mathsf{Full\text{-}Sel}$*)-secure, then* $\mathbf{CRFE}$ *is $2^{-\tau(\lambda)}\,\mathrm{negl}(\lambda)$-(*$\mathsf{Adap}$ *or* $\mathsf{Full\text{-}Sel}$*)-secure.*

We now can combine this with the following result from [GVW12].

Theorem 8 (1-Key Adap-Secure Public-Key FE for $\mathsf{P/poly}$ [GVW12])**.** *Let μ be any function from $\mathbb{N}$ to $[0,1]$. Assuming public key encryption with $\mu(\lambda)\,\mathrm{negl}(\lambda)$-security, there exist $\mu(\lambda)\,\mathrm{negl}(\lambda)$-*$\mathsf{Adap}$*-secure 1-key non-compact public-key FE schemes for* $\mathsf{P/poly}$.

Now, applying the transformation of Proposition 3 to the $\mu\,\mathrm{negl}$-Adap-secure 1-key FE schemes for $\mathsf{P/poly}$ with $\mu = 2^{-(\mathrm{i}\ell+\tau)}$, yields $2^{-\tau}\,\mathrm{negl}$-Adap-secure *collusion-resistant* (non-compact public-key) FE for $\mathsf{P/poly}$. Finally, note that it follows from [AJS15] that collusion-resistant non-compact FE schemes implies collusion-resistant compact FE schemes with the same level of security, which yields Theorem 4.

5.1 From 1-key to Collusion-Resistant FE, Generically

In this section, we prove Proposition 3, and give in particular an explicit transformation. Let us fix in particular any circuit class $\mathcal{C}$, any $\mathrm{i}\ell$ such that $\mathrm{i}\ell(\lambda) = \omega(\log \lambda) \leq \lambda$. The resulting collusion-resistant FE scheme for $\mathcal{C}$, denoted $\mathbf{CRFE} = (\mathsf{CRFE.Setup}, \mathsf{CRFE.KeyGen}, \mathsf{CRFE.Enc}, \mathsf{CRFE.Dec})$, then relies on the following building blocks.

- An $\mathrm{i}\ell$-bit-input indistinguishability obfuscator $i\mathcal{O}$ for $\mathsf{P/poly}$.
- A 1-key FE scheme $\mathbf{OFE} = (\mathsf{OFE.Setup}, \mathsf{OFE.KeyGen}, \mathsf{OFE.Enc}, \mathsf{OFE.Dec})$ for $\mathcal{C}$.
- A puncturable PRF scheme $\mathsf{PPRF} = (\mathsf{PRF.Gen}, \mathsf{PRF.Punc}, \mathsf{F})$.

Given the above building blocks, to construct collusion resistant FE **CRFE** for $\mathcal{C}$, we start with the following intuition. *If efficiency were not a problem*, we could trivially construct a FE scheme that support releasing any polynomial number of secret keys, essentially by using a super-polynomial number of instances of **OFE**. Concretely, we would proceed as follows:

- *Setup:* Genenerate a super-polynomial number, $M = 2^{\mathrm{i}\ell(\lambda)} = 2^{\omega(\lambda)}$, of **OFE** instances with master keys $\{(\mathsf{OMPK}_i, \mathsf{OMSK}_i) \xleftarrow{\$} \mathsf{OFE.Setup}(1^\lambda)\}_{i\in[M]}$.

- *Key Generation:* To generate a key for a function f, sample an index at random $i_f \xleftarrow{\$} [M]$ and generate a secret key using the i_f^{th} master secret key $\mathsf{OSK}_{i_f} \xleftarrow{\$} \mathsf{OFE.KeyGen}(\mathsf{OMSK}_{i_f}, f)$. Since there are at most a polynomial number of secret keys ever generated, the probability that every **OFE** instance is used to generate at most one secret key is overwhelming.

- *Encryption:* To encrypt any input x, simply encrypt the input x under all master public keys, $\{\mathsf{OCT}_i \xleftarrow{\$} \mathsf{OFE.Enc}(\mathsf{OMPK}_i, x)\}_{i\in[M]}$. Given the set of ciphertexts, one can compute the output $f(x)$ of any function f for which a secret key OSK_{i_f} has been generated, by decrypting the appropriated ciphertext OCT_{i_f} using the secret key OSK_{i_f}.

Of course, the only problem with this FE scheme is that its setup and encryption algorithms run in super-polynomial time. To address this, we follow the previously adopted idea (*e.g.* [BGL+15, CLTV15]) of using IO to "compress" these super-polynomially many **OFE** instances into "polynomial size". More precisely, instead of having the setup algorithm publish all M master public keys, let it generate an *obfuscated circuit* that on input $i \in [M]$ outputs the i^{th} master public key. Similarly, instead of having the encryption algorithm publish M ciphertexts, let it generate an obfuscated circuit that on input $i \in [M]$ outputs the i^{th} ciphertext under the i^{th} master public key. Since the inputs to the obfuscated circuits are indexes from the range $[M]$, which could be represented in $\mathrm{i}\ell$ bits, it suffices to use $\mathrm{i}\ell$-bit-input IO. Furthermore, for "compression" to the possible, all M master public and secret keys, as well as all M ciphertexts, need to be sampled using pseudo-randomness generated by puncturable PRFs. The resulting obfuscated circuits have polynomial size, since generating individual master public keys and ciphertexts using pseudorandomness is efficient, and hence the new FE scheme becomes efficient. Finally, the security of the new FE scheme follows from the common "one-input-at-a-time" argument, which incurs a $2^{-|i|} = 2^{-\mathrm{i}\ell}$ security loss. We formally describe the collusion-resistant FE scheme **CRFE** for $\mathcal{C}$ in Figure 5.

Next, we proceed to analyzing the correctness, efficiency, and security of the **CRFE** schemes.

Claim 2. *The scheme* **CRFE** *in Figure 5 is correct.*

Proof. The correctness of **CRFE** follows from the correctness of the underlying FE scheme **OFE** and the IO scheme $i\mathcal{O}$. Fix any λ, any function $f \in \mathcal{C}$, and any input x of f. Consider executing the algorithms of **CRFE** with f and x.

- *Setup:* An honestly generated master public key of **CRFE** is an obfuscated circuit of the program P_{setup}. By construction in Figure 6, P_{setup} on input any $i \in [M]$ outputs an honestly generated master public key OMPK_i of the underlying **OFE** scheme. Let $\mathsf{MPK} = \hat{P}_{\text{setup}}$. Then, by the correctness of the IO scheme, we have that $\hat{P}_{\text{setup}}(i)$ produces the master secret key OMPK_i.

- *Encryption:* A **CRFE** ciphertext CT of x is another obfuscated circuit of the program P_{enc}. By construction in Figure 7, P_{enc} on input any $i \in [M]$ outputs an honestly generated **OFE**

Collusion Resistant FE Scheme CRFE for $\mathcal{C}$

<u>SETUP:</u> $\mathsf{CRFE.Setup}(1^\lambda)$ does:

- Sample a PPRF key $K^s \xleftarrow{\$} \mathsf{PRF.Gen}(1^\lambda)$.
- Obfuscate the program $P_{\text{setup}}[0, K^s, \perp]$ described in Figure 6

$$\hat{P}_{\text{setup}} \xleftarrow{\$} i\mathcal{O}(1^\kappa, P_{\text{setup}}[0, K^s, \perp, \perp]) ,$$

where the IO scheme is invoked with a security parameter $\kappa = \max(\lambda, |P_{\text{setup}}|)$. (As shown in Claim 3 below, $|P_{\text{setup}}| = \text{poly}(\lambda)$.)

- Output $\mathsf{MPK} = \hat{P}_{\text{setup}}$ and $\mathsf{MSK} = K^s$.

<u>ENCRYPTION:</u> $\mathsf{CRFE.Enc}(\mathsf{MPK} = \hat{P}_{\text{setup}}, x)$ does the following to encrypt an input $x \in \{0,1\}^N$:

- Sample a PPRF key $K^e \xleftarrow{\$} \mathsf{PRF.Gen}(1^\lambda)$.
- Obfuscate the program $P_{\text{enc}}[\hat{P}_{\text{setup}}, 0, K^e, x, \perp, \perp]$ described in Figure 7,

$$\mathsf{CT} = \hat{P}_{\text{enc}} \xleftarrow{\$} i\mathcal{O}(1^{\kappa'}, P_{\text{enc}}[\hat{P}_{\text{setup}}, 0, K^e, x, \perp, \perp, \perp]) ,$$

where the IO scheme is invoked with a security parameter $\kappa' = \max(\lambda, |P_{\text{enc}}|)$. (As shown in Claim 3 below, $|P_{\text{enc}}| = \text{poly}(\lambda, N)$.)

- Output the obfuscated circuit as the ciphertext $\mathsf{CT} = \hat{P}_{\text{enc}}$.

<u>KEY GENERATION:</u> $\mathsf{CRFE.KeyGen}(\mathsf{MSK} = K^s, f)$ a key for function $f \in \mathcal{C}$ as follows:

- Sample at random an index $i_f \xleftarrow{\$} [M]$.
- Generate a secret key of f under the i_f^{th} master secret key,

$$\begin{aligned} (\mathsf{OMPK}_{i_f}, \mathsf{OMSK}_{i_f}) &= \mathsf{OFE.Setup}(1^\lambda \,;\, \mathsf{F}(K^s, i_f)) , \\ \mathsf{OSK}_{i_f} &\xleftarrow{\$} \mathsf{OFE.KeyGen}(\mathsf{OMSK}_{i_f}, f) . \end{aligned}$$

- Output $\mathsf{SK} = (i_f, \mathsf{OSK}_{i_f})$.

<u>DECRYPTION:</u> $\mathsf{CRFE.Dec}(\mathsf{SK} = (i_f, \mathsf{OSK}_{i_f}), \mathsf{CT} = \hat{P}_{\text{enc}})$ does:

- Compute the ciphertext of x under the i_f^{th} master public key,

$$\mathsf{OCT}_{i_f} = \hat{P}_{\text{enc}}(i_f) .$$

- Decrypt the obtained ciphertext using OSK_{i_f},

$$y = \mathsf{OFE.Dec}(\mathsf{OSK}_{i_f}, \mathsf{OCT}_{i_f}) .$$

- Output y.

Figure 5: Collusion Resistant FE Scheme **CRFE** for $\mathcal{C}$ from $\text{i}\ell(\lambda) = \omega(\lambda)$-bit-input IO

Circuit $P_{\text{setup}}[i^*, K^s, \mathsf{OMPK}^*]$

Constants: $i^* \in \{0, \cdots, M+1\}$ is an index, for $M = 2^{\mathrm{i}\ell(\lambda)}$ and $\mathrm{i}\ell = \omega(\log \lambda)$, K^s is a PPRF key, and OMPK^* is a master public key of the **OFE** scheme.

Input: Index $i \in [M]$.

Procedure:

1. If $i \neq i^*$, compute $(\mathsf{OMPK}_i, \mathsf{OMSK}_i) = \mathsf{OFE.Setup}(1^\lambda\ ;\ \mathsf{F}(K^s, i))$.
2. If $i = i^*$, output $\mathsf{OMPK}_{i^*} = \mathsf{OMPK}^*$.

Output OMPK_i.

Figure 6: Circuit P_{setup} in the construction and analysis of **CRFE**

Circuit $P_{\text{enc}}[\hat{P}_{\text{setup}}, i^*, K^e, x_0, x_1, \mathsf{OCT}^*]$

Constants: $\hat{P}_{\text{setup}}$ is an obfuscated program, $i^* \in \{0, \cdots, M+1\}$ is an index, for $M = 2^{\mathrm{i}\ell(\lambda)}$ and $\mathrm{i}\ell = \omega(\log \lambda)$, K^s is a PPRF key, $x_0, x_1 \in \{0,1\}^N$ are two inputs, and OCT^* is a ciphertext of **OFE**.

Input: Index $i \in [M]$.

Procedure:

1. If $i < i^*$, compute $\mathsf{OMPK}_i = \hat{P}_{\text{setup}}(i)$ and $\mathsf{OCT}_i = \mathsf{OFE.Enc}(\mathsf{OMPK}_i, \underline{x_1};\ \mathsf{F}(K^e, i))$.
2. If $i = i^*$, output $\mathsf{OCT}_{i^*} = \mathsf{OCT}^*$.
3. If $i > i^*$, compute $\mathsf{OMPK}_i = \hat{P}_{\text{setup}}(i)$ and $\mathsf{OCT}_i = \mathsf{OFE.Enc}(\mathsf{OMPK}_i, \underline{x_0};\ \mathsf{F}(K^e, i))$.

Output OCT_i.

Figure 7: Circuit P_{enc} in the construction and analysis of **CRFE**

ciphertext OCT_i of x under the key output by $\mathsf{MPK}(i) = \hat{P}_{\text{setup}}(i)$, which is OMPK_i as argued above. Let $\mathsf{CT} = \hat{P}_{\text{enc}}$. Then, by the correctness of the IO scheme, $\hat{P}_{\text{enc}}(i)$ produces the ciphertext OCT_i.

- *Key Generation:* A **CRFE** secret key SK of f contains a randomly chosen index $i_f \xleftarrow{\$} [M]$, and an **OFE** secret key OSK_{i_f} for f under the i_f^{th} master secret key MSK_{i_f}.

- *Decryption:* When decrypting CT using SK, the decryptor first evaluates $\hat{P}_{\text{enc}}(i_f)$ to obtain OCT_{i_f} as argued above. Next, the decryptor decrypts OCT_{i_f} using the secret key OSK_{i_f} contained in SK. By the correctness of the **OFE** scheme, the output is $f(x)$.

Therefore the scheme **CRFE** is correct. □

Next, we show that algorithms of the new scheme **CRFE** is only polynomially slower than that of **OFE**.

Claim 3. *There exists a universal polynomial p (independent of $\mathcal{C}$), such that, algorithms of* **CRFE** *in*

Figure 5 run in time,

$$\begin{aligned}
t_{\mathsf{CRFE.Setup}}(\lambda, N, D, S) &\leq p(\lambda,\ t_{\mathsf{OFE.Setup}}(\lambda, N, D, S)) \\
t_{\mathsf{CRFE.KeyGen}}(\lambda, N, D, S) &\leq p(\lambda,\ t_{\mathsf{OFE.KeyGen}}(\lambda, N, D, S)) \\
t_{\mathsf{CRFE.Enc}}(\lambda, N, D, S) &\leq p(\lambda,\ t_{\mathsf{OFE.Setup}}(\lambda, N, D, S), t_{\mathsf{OFE.Enc}}(\lambda, N, D, S)) \\
t_{\mathsf{CRFE.Dec}}(\lambda, N, D, S) &\leq p(\lambda,\ t_{\mathsf{OFE.Setup}}(\lambda, N, D, S), t_{\mathsf{OFE.Enc}}(\lambda, N, D, S), t_{\mathsf{OFE.Dec}}(\lambda, N, D, S))\ ,
\end{aligned}$$

where $t_\star(\lambda, N, S)$ denotes the runtime of algorithm $\star$, and $N = N(\lambda)$, $D = D(\lambda)$, $S = S(\lambda)$ are polynomial upper-bounds on the input-length, depth, and size of circuits that compute functions in $\mathcal{C}_\lambda$.

In particular, according the above claim, if the original FE scheme **OFE** is *compact*: There exists a polynomial q, such that, $t_{\mathsf{OFE.Setup}}(\lambda, N, D, S) \leq q(\lambda)$, $t_{\mathsf{OFE.Enc}}(\lambda, N, D, S) \leq q(\lambda, N)$, $t_{\mathsf{OFE.KeyGen}}(\lambda, N, D, S) \leq q(\lambda, S)$, and $t_{\mathsf{OFE.Dec}}(\lambda, N, D, S) \leq q(\lambda, S)$. Then the new FE scheme **CRFE** is also *compact* w.r.t. a different polynomial q'.

Proof. We first analyze the size of the programs P_{setup} and P_{enc}, which are obfuscated in **CRFE**. Observe that $P_{\text{setup}}(i)$ basically either invokes the setup algorithm of **OFE** with pseudo-randomness, or outputs a hardwired value. By the efficiency of the PPRF, we have that

$$|P_{\text{setup}}| = \text{poly}(t_{\mathsf{OFE.Setup}}(\lambda, N, D, S))\ .$$

Furthermore, by the efficiency of the IO scheme, an obfuscation $\hat{P}_{\text{setup}}$ of P_{setup} has size

$$\begin{aligned}
|\hat{P}_{\text{setup}}| &\leq \mathsf{Time}_{i\mathcal{O}}(1^\kappa, P_{\text{setup}}) \\
&= \text{poly}(\kappa, |P_{\text{setup}}|) \\
&= \text{poly}(\max(\lambda, |P_{\text{setup}}|),\ |P_{\text{setup}}|) \\
&= \text{poly}(t_{\mathsf{OFE.Setup}}(\lambda, N, D, S))\ .
\end{aligned}$$

Next, observe that $P_{\text{enc}}(i)$ in figure 6 basically either evaluates $\mathsf{OMPK}_i = \hat{P}_{\text{setup}}(i)$ and generates a ciphertext of the input x under OMPK_i, or outputs a hardwired ciphertext. By the above analysis of the size of $\hat{P}_{\text{setup}}$, and the efficiency of PPRF, we have that

$$|P_{\text{enc}}| = \text{poly}(t_{\mathsf{OFE.Setup}}(\lambda, N, D, S),\ t_{\mathsf{OFE.Enc}}(\lambda, N, D, S))\ .$$

Next, we analyze the efficiency of the algorithms of **CRFE**.

- *Setup:* The runtime of the setup algorithm is dominated by the step of obfuscating the program P_{setup}, that is,

$$\begin{aligned}
t_{\mathsf{CRFE.Setup}}(\lambda, N, D, S) &= \mathsf{Time}_{i\mathcal{O}}(1^\kappa, P_{\text{setup}}) + \text{poly}(\lambda) \\
&= \text{poly}(\kappa, |P_{\text{setup}}|) + \text{poly}(\lambda) \\
&= \text{poly}(\max(\lambda, |P_{\text{setup}}|), |P_{\text{setup}}|) + \text{poly}(\lambda) \\
&\leq p(\lambda,\ t_{\mathsf{OFE.Setup}}(\lambda, N, D, S))\ ,
\end{aligned}$$

  where the second equality follows from the efficiency of the IO scheme (and the last inequality holds when p is sufficiently large).

- *Encryption:* The runtime of the encryption algorithm is dominated by obfuscating the program P_{enc}, that is,

$$\begin{aligned} t_{\mathsf{CRFE.Enc}}(\mathsf{MPK}, x) &= \mathsf{Time}_{i\mathcal{O}}(1^{\kappa'}, P_{\text{enc}}) + \text{poly}(\lambda) \\ &= \text{poly}(\kappa', |P_{\text{enc}}|) + \text{poly}(\lambda) \\ &= \text{poly}(\max(\lambda, |P_{\text{enc}}|), |P_{\text{enc}}|) + \text{poly}(\lambda) \\ &\leq p(\lambda,\ \lambda,\ t_{\mathsf{OFE.Setup}}(\lambda), t_{\mathsf{OFE.Enc}}(\lambda, N, S)) \ . \end{aligned}$$

- *Key Generation:* The runtime of the key generation algorithm is dominated by generating a secret key of the $\mathbf{OFE}$ scheme. More precisely,

$$\begin{aligned} t_{\mathsf{CRFE.KeyGen}}(\mathsf{MSK}, f) &= t_{\mathsf{OFE.KeyGen}}(\lambda, N, S) + \text{poly}(\lambda) \\ &\leq p(\lambda,\ t_{\mathsf{OFE.KeyGen}}(\lambda, N, S)) \ . \end{aligned}$$

- *Decryption:* The decryption algorithm involves evaluating the obfuscated circuit $\hat{P}_{\text{enc}}$ contained in the ciphertext CT on input i_f, and decrypting the obtained $\mathbf{OFE}$ ciphertext OCT_{i_f} using the $\mathbf{OFE}$ secret key OSK_{i_f} contained in the secret key SK. Therefore,

$$\begin{aligned} t_{\mathsf{CRFE.Dec}}(\mathsf{CT}, \mathsf{SK}) &= |\hat{P}_{\text{enc}}| + t_{\mathsf{OFE.Dec}}(\lambda, N, S) + \text{poly}(\lambda) \\ &\leq p(\lambda,\ t_{\mathsf{OFE.Setup}}(\lambda), t_{\mathsf{OFE.Enc}}(\lambda, N, S), t_{\mathsf{OFE.Dec}}(\lambda, N, S)) \end{aligned}$$

This concludes the claim. □

Lemma 9. *If $i\mathcal{O}$ and PPRF are $2^{-(\mathrm{i}\ell(\lambda)+\tau(\lambda))}\,\mathrm{negl}(\lambda)$-indistinguishable, and $\mathbf{OFE}$ is $2^{-(\mathrm{i}\ell(\lambda)+\tau(\lambda))}\,\mathrm{negl}(\lambda)$-(*Adap *or* Full-Sel*)-secure, then, $\mathbf{CRFE}$ in Figure 5 is $2^{-\tau(\lambda)}\,\mathrm{negl}(\lambda)$-(*Adap *or* Full-Sel*)-secure.*

Proof. We prove the theorem for the case of Adap-security; the proof for the case of Full-Sel-security are syntactically identical.

Fix any PPT attacker A, we need to show that the advantage of A in games $\mathsf{IND}_A^{\mathbf{CRFE}}(1^\lambda, 0)$ and $\mathsf{IND}_A^{\mathbf{CRFE}}(1^\lambda, 1)$ is bounded by $2^{-\tau}\,\mathrm{negl}$.

$$\mathsf{Advt}_A^{\mathbf{CRFE}} = \left| \Pr[\mathsf{IND}_A^{\mathbf{CRFE}}(1^\lambda, 0) = 1] - \Pr[\mathsf{IND}_A^{\mathbf{CRFE}}(1^\lambda, 1) = 1] \right| \leq 2^{-\tau(\lambda)}\,\mathrm{negl}(\lambda)$$

Recall that the game $\mathsf{IND}_A^{\mathbf{CRFE}}(1^\lambda, b)$ proceeds in four stages: 1) The challenger samples a pair of master keys $(\mathsf{MPK}, \mathsf{MSK}) \xleftarrow{\$} \mathsf{CRFE.Setup}(1^\lambda)$ and sends MPK to A; 2) A can obtain an arbitrary number of secret keys $\{\mathsf{SK}_i\}$ for functions $\{f_i\}$ it chooses adaptively; 3) A chooses two challenge messages (x_0, x_1) and receives the ciphertext CT of x_b; 4) A again obtains secret keys of functions of its choice.

To bound the advantage of A, we define hybrids $\left\{H_{i^*}^0, \cdots, H_{i^*}^3\right\}_{i^* \in [M+1]}$, and show that the advantage of A in distinguishing $H_{i^*}^j$ from $H_{i^*}^{j+1}$ for any $0 \leq j \leq 2$, as well as in distinguishing $H_{i^*}^3$ and $H_{i^*+1}^0$, is bounded by $\mu(\lambda) = 2^{-(\mathrm{i}\ell(\lambda)+\tau(\lambda))}\,\mathrm{negl}(\lambda)$. Furthermore, the advantage of A in distinguishing H_0^0 from $\mathsf{IND}_A^{\mathbf{CRFE}}(1^\lambda, 0)$, and H_{M+1}^0 from $\mathsf{IND}_A^{\mathbf{CRFE}}(1^\lambda, 1)$ is also bounded by μ. Next, we formally describe the hybrids.

Hybrid $H_{i^*}^0$: This hybrid proceeds identically to $\mathsf{IND}_A^{\mathbf{CRFE}}(1^\lambda, b)$, except that, the master public key MPK and ciphertext CT are generated differently. Recall that by construction of $\mathbf{CRFE}$, the master public key and ciphertext are obfuscated circuits of programs P_{setup} and P_{enc} respectively.

$$\begin{aligned}\mathsf{MPK} &= \hat{P}_{\text{setup}} \xleftarrow{\$} i\mathcal{O}(1^\kappa, P_{\text{setup}}) \\ \mathsf{CT} &= \hat{P}_{\text{enc}} \xleftarrow{\$} i\mathcal{O}(1^{\kappa'}, P_{\text{enc}})\end{aligned}$$

In $\mathsf{IND}_A^{\mathbf{CRFE}}(1^\lambda, b)$, the two programs are $P_{\text{setup}} = P_{\text{setup}}[0, K^s, \perp]$ and $P_{\text{enc}} = P_{\text{enc}}[\hat{P}_{\text{setup}}, 0, K^e, x_b, \perp, \perp]$, where $\hat{P}_{\text{setup}}$ is the obfuscation of P_{setup}.

In $H_{i^*}^0$, the master public key and ciphertext contain obfuscation of P_{setup} and P_{enc} hardwired with different constants as described below (recall that $K\{i\}$ denotes a PPRF key punctured at point i).

$$\begin{aligned}P_{\text{setup}} &= P_{\text{setup}}[i^*, K^s\{i^*\}, \mathsf{OMPK}_{i^*}] \ , \\ &\quad \text{where } (\mathsf{OMPK}_{i^*}, \mathsf{OMSK}_{i^*}) \xleftarrow{\$} \mathsf{OFE.Setup}(1^\lambda \ ; \ \mathsf{F}(K^s \ , \ i^*)) \ , \text{ and} \qquad (12) \\ P_{\text{enc}} &= P_{\text{enc}}[\hat{P}_{\text{setup}}, i^*, K^e\{i^*\}, x_0, x_1, \mathsf{OCT}_{i^*}] \ , \\ &\quad \text{where } \mathsf{OCT}_{i^*} \xleftarrow{\$} \mathsf{OFE.Enc}(\mathsf{OMPK}_{i^*}, x_0 \ ; \ \mathsf{F}(K^e \ , \ i^*)) \ . \qquad (13)\end{aligned}$$

The rest of the experiment proceeds identically to $\mathsf{IND}_A^{\mathbf{CRFE}}(1^\lambda, b)$. Finally, $H_{i^*}^0$ outputs the bit that A outputs.

Below, we show that

Claim 4. *If $i\mathcal{O}$ is μ-secure, then the outputs of $\mathsf{IND}_A^{\mathbf{CRFE}}(1^\lambda, 0)$ and H_0^0, and the outputs of $\mathsf{IND}_A^{\mathbf{CRFE}}(1^\lambda, 1)$ and H_{M+1}^0, are μ-close.*

Proof. By definition of $P_{\text{setup}}[\star]$ and $P_{\text{enc}}[\star]$ in Figure 6 and 7 and the correctness of PPRF, we have that the programs obfuscated in $\mathsf{IND}_A^{\mathbf{CRFE}}(1^\lambda, 0)$ are functionally equivalent to that obfuscated in H_0^0, that is,

$$\begin{aligned}P_{\text{setup}}[0, K^s, \perp] &\equiv P_{\text{setup}}[0, K^s\{0\}, \mathsf{OMPK}_0] \\ P_{\text{enc}}[\hat{P}_{\text{setup}}, 0, K^e, x_0, \perp, \perp] &\equiv P_{\text{enc}}[\hat{P}_{\text{setup}}, 0, K^e\{0\}, x_0, x_1, \mathsf{OCT}_0]\end{aligned}$$

where OMSK_0 and OCT_0 are generated as in Equation (12) and (13). Therefore, if $i\mathcal{O}$ is μ-indistinguishable, the outputs of $\mathsf{IND}_A^{\mathbf{CRFE}}(1^\lambda, 0)$ and H_0^0 are μ-close.

Similarly, the outputs of $\mathsf{IND}_A^{\mathbf{CRFE}}(1^\lambda, 1)$ and H_{M+1}^0 are also μ-close, following from the fact that the programs obfuscated in them are functionally equivalent.

$$\begin{aligned}P_{\text{setup}}[0, K^s, \perp] &\equiv P_{\text{setup}}[M+1, K^s\{M+1\}, \mathsf{OMPK}_{M+1}] \\ P_{\text{enc}}[\hat{P}_{\text{setup}}, 0, K^e, x_1, \perp, \perp] &\equiv P_{\text{enc}}[\hat{P}_{\text{setup}}, M+1, K^e\{M+1\}, x_0, x_1, \mathsf{OCT}_{M+1}]\end{aligned}$$

□

Hybrid $H_{i^*}^1$: This hybrid proceed identically to $H_{i^*}^0$ except that the master public key OMPK_{i^*} and ciphertext OCT_{i^*} hardwired in P_{setup} and P_{enc} are generated using true randomness, instead of pseudorandomness generated by PPRF.

$$\begin{aligned}(\mathsf{OMPK}_{i^*}, \mathsf{OMSK}_{i^*}) &\xleftarrow{\$} \mathsf{OFE.Setup}(1^\lambda \ ; \ \underline{U_{\text{poly}(\lambda)}}) \ , \text{ and} \\ \mathsf{OCT}_{i^*} &\xleftarrow{\$} \mathsf{OFE.Enc}(\mathsf{OMPK}_{i^*}, x_0 \ ; \ \underline{U_{\text{poly}(\lambda)}}) \ .\end{aligned}$$

The rest of the experiment proceeds identically to $H^0_{i^*}$.

Since the only difference between $H^0_{i^*}$ and $H^1_{i^*}$ lies in whether OMPK_{i^*} and OCT_{i^*} are generated by pseudorandomness or true randomness, it follows from the μ-indistinguishability of PPRF that the outputs of $H^0_{i^*}$ and $H^0_{i^*}$ are μ-close.

Claim 5. *If PPRF is $\mu(\lambda)$-secure, then the outputs of $H^0_{i^*}$ and $H^0_{i^*}$ are $\mu(\lambda)$-close.*

Hybrid $H^2_{i^*}$: This hybrid proceeds identically to $H^1_{i^*}$ except that the ciphertext OCT_{i^*} hardwired P_{enc} encrypts x_1, instead of x_0.

$$\begin{aligned}(\mathsf{OMPK}_{i^*}, \mathsf{OMSK}_{i^*}) &\xleftarrow{\$} \mathsf{OFE.Setup}(1^\lambda\ ;\ U_{\mathrm{poly}(\lambda)})\ , \text{ and}\\ \mathsf{OCT}_{i^*} &\xleftarrow{\$} \mathsf{OFE.Enc}(\mathsf{OMPK}_{i^*}, \underline{x_1}\ ;\ U_{\mathrm{poly}(\lambda)})\ .\end{aligned}$$

Note that the only difference between $H^1_{i^*}$ and $H^2_{i^*}$ lies in whether OCT_{i^*} encrypts x_0 or x_1. In both hybrids, the $(i^*)^{\text{th}}$ master keys $(\mathsf{OMPK}_{i^*}, \mathsf{OMSK}_{i^*})$ and OCT_{i^*} are generated honestly using true randomness. By construction of **CRFE**, the secret key SK of a function f contains an **OFE** secret key OSK_{i_f} of f under a randomly chosen master secret key OMPK_{i_f} from a super-polynomial number M of master secret keys $i_f \xleftarrow{\$} [M]$. Since the attacker A obtains at most a polynomial number of secret keys, the probability that OMSK_{i^*} is used to generate two secret key is negligible. Conditioned on this event happening, it follows from the μ-Adap-security of **OFE** that the ciphertext OCT_{i^*} in $H^1_{i^*}$ encrypting x_0 is indistinguishable to that in $H^2_{i^*}$ encrypting x_1. Therefore,

Claim 6. *If* **OFE** *is $\mu(\lambda)$-Adap-secure, then the outputs of $H^1_{i^*}$ and $H^2_{i^*}$ are $\mu(\lambda)$-close.*

Hybrid $H^3_{i^*}$: This hybrid proceeds identically to $H^2_{i^*}$ except that the master public key OMPK_{i^*} and ciphertext OCT_{i^*} hardwired in P_{setup} and P_{enc} are generated using pseudorandomness generated by PPRF, instead of true randomness.

$$\begin{aligned}(\mathsf{OMPK}_{i^*}, \mathsf{OMSK}_{i^*}) &\xleftarrow{\$} \mathsf{OFE.Setup}(1^\lambda\ ;\ \underline{\mathsf{PPRF}(K^s\ ,\ i^*)})\ , \text{ and} &(14)\\ \mathsf{OCT}_{i^*} &\xleftarrow{\$} \mathsf{OFE.Enc}(\mathsf{OMPK}_{i^*}, x_1\ ;\ \underline{\mathsf{PPRF}(K^e\ ,\ i^*)})\ . &(15)\end{aligned}$$

It again follows from the μ-indistinguishability of PPRF that

Claim 7. *If PPRF is $\mu(\lambda)$-indistinguishable, then the outputs of $H^2_{i^*}$ and $H^3_{i^*}$ are $\mu(\lambda)$-close.*

Furthermore, we note that for every $i^* \in [M]$, the programs P_{setup} and P_{enc} obfuscated in hybrids $H^3_{i^*}$ and $H^0_{i^*+1}$ are functionally equivalent.

$$\begin{aligned}P_{\mathrm{setup}}[i^*, K^s\{i^*\}, \mathsf{OMPK}_{i^*}] &\equiv P_{\mathrm{setup}}[i^*+1, K^s\{i^*+1\}, \mathsf{OMPK}_{i^*+1}]\\ P_{\mathrm{enc}}[\hat{P}_{\mathrm{setup}}, i^*, K^e\{i^*\}, x_0, x_1, \mathsf{OCT}_{i^*}] &\equiv P_{\mathrm{enc}}[\hat{P}_{\mathrm{setup}}, i^*+1, K^e\{i^*+1\}, x_0, x_1, \mathsf{OCT}_{i^*+1}]\end{aligned}$$

where on the left hand side of the equations, the hardwired master public key OMPK_{i^*} and ciphertext OCT_{i^*} are generated as in Equation (14) and (15), while on the right hand side, the mater public key OMPK_{i^*+1} and ciphertext OCT_{i^*+1} are generated as in Equation (12) and (13). By definition of $P_{\mathrm{setup}}[\star]$, the P_{setup} programs with different hardwired constants described above are functionally equivalent because for any input $i \in [M]$, they both output

OMPK_i generated honestly using $\mathsf{OFE.Setup}$ with pseudorandomness $\mathsf{F}(K^s, i)$. By definition of $P_{\text{enc}}[\star]$, the P_{enc} programs above are also functionally equivalent because for any input $i \in [M]$ and $i \leq i^*$, they both produce OCT_i encrypting x_1 using pseudorandomness $\mathsf{F}(K^e, i)$, and for any $i \in [M]$ and $i > i^*$, they both produce OCT_i encrypting x_0 using pseudorandomness $\mathsf{F}(K^e, i)$. Therefore, if follows from the μ-indistinguishability of $i\mathcal{O}$ that the outputs of $H^3_{i^*}$ and $H^0_{i^*+1}$ are μ-close.

Claim 8. *If $i\mathcal{O}$ is μ-secure, then the outputs of $H^3_{i^*}$ and $H^0_{i^*+1}$ are μ-close.*

Using the above hybrids, we now conclude the lemma. Since there are in total $O(M)$ hybrids, it then follows from a hybrid argument that the advantage of A in distinguishing $\mathsf{IND}^{\mathbf{CRFE}}_A(1^\lambda, 0)$ and $\mathsf{IND}^{\mathbf{CRFE}}_A(1^\lambda, 1)$ is bounded by $O(M) \times \mu = 2^{-\tau(\lambda)}\,\mathrm{negl}(\lambda)$. □

Acknowledgements

The authors thank Benny Applebaum and Vinod Vaikuntanathan for many helpful discussions and insights.

References

[AB15] Benny Applebaum and Zvika Brakerski. Obfuscating circuits via composite-order graded encoding. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015, Part II*, volume 9015 of *LNCS*, pages 528–556, Warsaw, Poland, March 23–25, 2015. Springer, Heidelberg, Germany.

[ABCP15] Michel Abdalla, Florian Bourse, Angelo De Caro, and David Pointcheval. Simple functional encryption schemes for inner products. In Jonathan Katz, editor, *Public-Key Cryptography - PKC 2015 - 18th IACR International Conference on Practice and Theory in Public-Key Cryptography, Gaithersburg, MD, USA, March 30 - April 1, 2015, Proceedings*, volume 9020 of *Lecture Notes in Computer Science*, pages 733–751. Springer, 2015.

[ABR16] Benny Applebaum, Andrej Bogdanov, and Alon Rosen. A dichotomy for local small-bias generators. *Journal of Cryptology*, 29(3):577–596, July 2016.

[ABSV15] Prabhanjan Ananth, Zvika Brakerski, Gil Segev, and Vinod Vaikuntanathan. From selective to adaptive security in functional encryption. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 657–677, Santa Barbara, CA, USA, August 16–20, 2015. Springer, Heidelberg, Germany.

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

[AIK04] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Cryptography in nc^0. In *FOCS*, 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 Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part I*, volume 9215 of *LNCS*, pages 308–326, Santa Barbara, CA, USA, August 16–20, 2015. Springer, Heidelberg, Germany.

[AJS15] Prabhanjan Ananth, Abhishek Jain, and Amit Sahai. Achieving compactness generically: Indistinguishability obfuscation from non-compact functional encryption. *IACR Cryptology ePrint Archive*, 2015:730, 2015.

[AL16] Benny Applebaum and Shachar Lovett. Algebraic attacks against random local functions and their countermeasures. In Daniel Wichs and Yishay Mansour, editors, *48th ACM STOC*, pages 1087–1100, Cambridge, MA, USA, June 18–21, 2016. ACM Press.

[App12] Benny Applebaum. Pseudorandom generators with long stretch and low locality from random local one-way functions. In Howard J. Karloff and Toniann Pitassi, editors, *44th ACM STOC*, pages 805–816, New York, NY, USA, May 19–22, 2012. ACM Press.

[App15] Benny Applebaum. The cryptographic hardness of random local functions – survey. Cryptology ePrint Archive, Report 2015/165, 2015. http://eprint.iacr.org/2015/165.

[AR16] Benny Applebaum and Pavel Raykov. Fast pseudorandom functions based on expander graphs. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B, Part I*, volume 9985 of *LNCS*, pages 27–56, Beijing, China, October 31 – November 3, 2016. Springer, Heidelberg, Germany.

[AS16] Prabhanjan Ananth and Amit Sahai. Projective arithmetic functional encryption and indistinguishability obfuscation from degree-5 multilinear maps. Cryptology ePrint Archive, Report 2016/1097, 2016. http://eprint.iacr.org/2016/1097.

[BCFG17] Carmen Elisabetta Zaira Baltico, Dario Catalano, Dario Fiore, and Romain Gay. Practical functional encryption for quadratic functions with applications to predicate encryption. Cryptology ePrint Archive, Report 2017/151, 2017. http://eprint.iacr.org/2017/151.

[BGI+01a] Boaz Barak, Oded Goldreich, Rusell Impagliazzo, Steven Rudich, Amit Sahai, Salil Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In *Advances in Cryptology CRYPTO 2001*, pages 1–18. Springer, 2001.

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

[BGI14] Elette Boyle, Shafi Goldwasser, and Ioana Ivan. Functional signatures and pseudorandom functions. In *PKC*, pages 501–519, 2014.

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

[BGL+15] Nir Bitansky, Sanjam Garg, Huijia Lin, Rafael Pass, and Sidharth Telang. Succinct randomized encodings and their applications. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 439–448. ACM, 2015.

[BJK15] Allison Bishop, Abhishek Jain, and Lucas Kowalczyk. Function-hiding inner product encryption. In Tetsu Iwata and Jung Hee Cheon, editors, *Advances in Cryptology - ASIACRYPT 2015 - 21st International Conference on the Theory and Application of Cryptology and Information Security, Auckland, New Zealand, November 29 - December 3, 2015, Proceedings, Part I*, volume 9452 of *Lecture Notes in Computer Science*, pages 470–491. Springer, 2015.

[BNPW16] Nir Bitansky, Ryo Nishimaki, Alain Passelègue, and Daniel Wichs. From cryptomania to obfustopia through secret-key functional encryption. In Martin Hirt and Adam D. Smith, editors, *Theory of Cryptography - 14th International Conference, TCC 2016-B, Beijing, China, October 31 - November 3, 2016, Proceedings, Part II*, volume 9986 of *Lecture Notes in Computer Science*, pages 391–418, 2016.

[BPR15] Nir Bitansky, Omer Paneth, and Alon Rosen. On the cryptographic hardness of finding a nash equilibrium. In Guruswami [Gur15], pages 1480–1498.

[BQ12] Andrej Bogdanov and Youming Qiao. On the security of goldreich's one-way function. *Computational Complexity*, 21(1):83–127, 2012.

[BR14] Zvika Brakerski and Guy N. Rothblum. Virtual black-box obfuscation for all circuits via generic graded encoding. In Yehuda Lindell, editor, *TCC 2014*, volume 8349 of *LNCS*, pages 1–25, San Diego, CA, USA, February 24–26, 2014. Springer, Heidelberg, Germany.

[BS02] Dan Boneh and Alice Silverberg. Applications of multilinear forms to cryptography. *IACR Cryptology ePrint Archive*, 2002:80, 2002.

[BSW06] Dan Boneh, Amit Sahai, and Brent Waters. Fully collusion resistant traitor tracing with short ciphertexts and private keys. In Serge Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 573–592, St. Petersburg, Russia, May 28 – June 1, 2006. Springer, Heidelberg, Germany.

[BSW12] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: a new vision for public-key cryptography. *Commun. ACM*, 55(11):56–64, 2012.

[BV15] Nir Bitansky and Vinod Vaikuntanathan. Indistinguishability obfuscation from functional encryption. In *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015*, pages 171–190, 2015.

[BW13] Dan Boneh and Brent Waters. Constrained pseudorandom functions and their applications. In *ASIACRYPT (2)*, pages 280–300, 2013.

[CEMT09] James Cook, Omid Etesami, Rachel Miller, and Luca Trevisan. Goldreich's one-way function candidate and myopic backtracking algorithms. In Omer Reingold, editor, *TCC 2009*, volume 5444 of *LNCS*, pages 521–538. Springer, Heidelberg, Germany, March 15–17, 2009.

[CFN94] Benny Chor, Amos Fiat, and Moni Naor. Tracing traitors. In Yvo Desmedt, editor, *CRYPTO'94*, volume 839 of *LNCS*, pages 257–270, Santa Barbara, CA, USA, August 21–25, 1994. Springer, Heidelberg, Germany.

[CGH+15] Jean-Sébastien Coron, Craig Gentry, Shai Halevi, Tancrède Lepoint, Hemanta K. Maji, Eric Miles, Mariana Raykova, Amit Sahai, and Mehdi Tibouchi. Zeroizing without low-level zeroes: New MMAP attacks and their limitations. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part I*, volume 9215 of *LNCS*, pages 247–266, Santa Barbara, CA, USA, August 16–20, 2015. Springer, Heidelberg, Germany.

[CGH16] Yilei Chen, Craig Gentry, and Shai Halevi. Cryptanalyses of candidate branching program obfuscators. Cryptology ePrint Archive, Report 2016/998, 2016. `http://eprint.iacr.org/2016/998`.

[CHL+15] Jung Hee Cheon, Kyoohyung Han, Changmin Lee, Hansol Ryu, and Damien Stehlé. Cryptanalysis of the multilinear map over the integers. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part I*, volume 9056 of *LNCS*, pages 3–12, Sofia, Bulgaria, April 26–30, 2015. Springer, Heidelberg, Germany.

[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 Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 476–493, Santa Barbara, CA, USA, August 18–22, 2013. Springer, Heidelberg, Germany.

[CLT15] Jean-Sébastien Coron, Tancrède Lepoint, and Mehdi Tibouchi. New multilinear maps over the integers. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part I*, volume 9215 of *LNCS*, pages 267–286, Santa Barbara, CA, USA, August 16–20, 2015. Springer, Heidelberg, Germany.

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

[CM01] M. Cryan and P. B. Miltersen. On pseudorandom generators in nc0. In Proc. 26th MFCS, 2001.

[DIJK09] Yevgeniy Dodis, Russell Impagliazzo, Ragesh Jaiswal, and Valentine Kabanets. Security amplification for interactive cryptographic primitives. In Omer Reingold, editor, *TCC 2009*, volume 5444 of *LNCS*, pages 128–145. Springer, Heidelberg, Germany, March 15–17, 2009.

[DNR+09] Cynthia Dwork, Moni Naor, Omer Reingold, Guy N. Rothblum, and Salil P. Vadhan. On the complexity of differentially private data release: efficient algorithms and hardness results. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*, pages 381–390, 2009.

[DS05] Yevgeniy Dodis and Adam Smith. Correcting errors without leaking partial information. In Harold N. Gabow and Ronald Fagin, editors, *37th ACM STOC*, pages 654–663, Baltimore, MA, USA, May 22–24, 2005. ACM Press.

[Gay16] Romain Gay. Functional encryption for quadratic functions, and applications to predicate encryption. Cryptology ePrint Archive, Report 2016/1106, 2016. http://eprint.iacr.org/2016/1106.

[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 Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015, Part II*, volume 9015 of *LNCS*, pages 498–527, Warsaw, Poland, March 23–25, 2015. Springer, Heidelberg, Germany.

[GGHZ16] Sanjam Garg, Craig Gentry, Shai Halevi, and Mark Zhandry. Functional encryption without obfuscation. In Eyal Kushilevitz and Tal Malkin, editors, *TCC 2016-A, Part II*, volume 9563 of *LNCS*, pages 480–511, Tel Aviv, Israel, January 10–13, 2016. Springer, Heidelberg, Germany.

[GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *J. ACM*, 33(4):792–807, 1986.

[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, *45th ACM STOC*, pages 555–564, Palo Alto, CA, USA, June 1–4, 2013. ACM Press.

[GLSW15] Craig Gentry, Allison Bishop Lewko, Amit Sahai, and Brent Waters. Indistinguishability obfuscation from the multilinear subgroup elimination assumption. In Guruswami [Gur15], pages 151–170.

[GMS16] Sanjam Garg, Pratyay Mukherjee, and Akshayaram Srinivasan. Obfuscation without the vulnerabilities of multilinear maps. *IACR Cryptology ePrint Archive*, 2016:390, 2016.

[GNW11] Oded Goldreich, Noam Nisan, and Avi Wigderson. *On Yao's XOR-Lemma*, pages 273–301. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[Gol00] Oded Goldreich. Candidate one-way functions based on expander graphs. *Electronic Colloquium on Computational Complexity (ECCC)*, 7(90), 2000.

[Gol01] Oded Goldreich. *Foundations of Cryptography — Basic Tools*. Cambridge University Press, 2001.

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

[GPSZ16] Sanjam Garg, Omkant Pandey, Akshayaram Srinivasan, and Mark Zhandry. Breaking the sub-exponential barrier in obfustopia. Cryptology ePrint Archive, Report 2016/102, 2016. http://eprint.iacr.org/2016/102.

[Gur15] Venkatesan Guruswami, editor. *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015*. IEEE Computer Society, 2015.

[GVW12] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Functional encryption with bounded collusions via multi-party computation. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 162–179, Santa Barbara, CA, USA, August 19–23, 2012. Springer, Heidelberg, Germany.

[IK02] Yuval Ishai and Eyal Kushilevitz. Perfect constant-round secure computation via perfect randomizing polynomials. In *ICALP*, pages 244–256, 2002.

[KPTZ13] Aggelos Kiayias, Stavros Papadopoulos, Nikos Triandopoulos, and Thomas Zacharias. Delegatable pseudorandom functions and applications. In *CCS*, pages 669–684, 2013.

[KS17] Ilan Komargodski and Gil Segev. From minicrypt to obfustopia via private-key functional encryption. Cryptology ePrint Archive, Report 2017/080, 2017. http://eprint.iacr.org/2017/080.

[Lin16a] Huijia Lin. Indistinguishability obfuscation from constant-degree graded encoding schemes. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part I*, volume 9665 of *LNCS*, pages 28–57, Vienna, Austria, May 8–12, 2016. Springer, Heidelberg, Germany.

[Lin16b] Huijia Lin. Indistinguishability obfuscation from ddh on 5-linear maps and locality-5 prgs. Cryptology ePrint Archive, Report 2016/1096, 2016. http://eprint.iacr.org/2016/1096.

[LPST16] Huijia Lin, Rafael Pass, Karn Seth, and Sidharth Telang. Output-compressing randomized encodings and applications. In Eyal Kushilevitz and Tal Malkin, editors, *TCC 2016-A, Part I*, volume 9562 of *LNCS*, pages 96–124, Tel Aviv, Israel, January 10–13, 2016. Springer, Heidelberg, Germany.

[LSS14] Adeline Langlois, Damien Stehlé, and Ron Steinfeld. Gghlite: More efficient multilinear maps from ideal lattices. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology - EUROCRYPT 2014 - 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings*, volume 8441 of *Lecture Notes in Computer Science*, pages 239–256. Springer, 2014.

[LV16] Huijia Lin and Vinod Vaikuntanathan. Indistinguishability obfuscation from ddh-like assumptions on constant-degree graded encodings. In *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, New Brunswick, NJ, USA, 9-11 October, 2016*, 2016.

[MST03] Elchanan Mossel, Amir Shpilka, and Luca Trevisan. On e-biased generators in NC0. In *44th FOCS*, pages 136–145, Cambridge, MA, USA, October 11–14, 2003. IEEE Computer Society Press.

[MSZ16a] Eric Miles, Amit Sahai, and Mark Zhandry. Annihilation attacks for multilinear maps: Cryptanalysis of indistinguishability obfuscation over GGH13. *IACR Cryptology ePrint Archive*, 2016:147, 2016.

[MSZ16b] Eric Miles, Amit Sahai, and Mark Zhandry. Annihilation attacks for multilinear maps: Cryptanalysis of indistinguishability obfuscation over GGH13. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part II*, volume 9815 of *LNCS*, pages 629–658, Santa Barbara, CA, USA, August 14–18, 2016. Springer, Heidelberg, Germany.

[MT10] Ueli M. Maurer and Stefano Tessaro. A hardcore lemma for computational indistinguishability: Security amplification for arbitrarily weak PRGs with optimal stretch. In Daniele Micciancio, editor, *TCC 2010*, volume 5978 of *LNCS*, pages 237–254, Zurich, Switzerland, February 9–11, 2010. Springer, Heidelberg, Germany.

[O'N10] Adam O'Neill. Definitional issues in functional encryption. Cryptology ePrint Archive, Report 2010/556, 2010. http://eprint.iacr.org/.

[OW14] Ryan O'Donnell and David Witmer. Goldreich's PRG: evidence for near-optimal polynomial stretch. In *IEEE 29th Conference on Computational Complexity, CCC 2014, Vancouver, BC, Canada, June 11-13, 2014*, pages 1–12, 2014.

[PRV12] Bryan Parno, Mariana Raykova, and Vinod Vaikuntanathan. How to delegate and verify in public: Verifiable computation from attribute-based encryption. In Ronald Cramer, editor, *TCC 2012*, volume 7194 of *LNCS*, pages 422–439, Taormina, Sicily, Italy, March 19–21, 2012. Springer, Heidelberg, Germany.

[PST14] Rafael Pass, Karn Seth, and Sidharth Telang. Indistinguishability obfuscation from semantically-secure multilinear encodings. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 500–517, Santa Barbara, CA, USA, August 17–21, 2014. Springer, Heidelberg, Germany.

[SW14] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: Deniable encryption, and more. *Proc. of STOC 2014*, 2014.

[Ull13] Jonathan Ullman. Answering $n^{2+o(1)}$ counting queries with differential privacy is hard. In *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 361–370, 2013.

[Yao82] Andrew Chi-Chih Yao. Theory and applications of trapdoor functions (extended abstract). In *23rd FOCS*, pages 80–91, Chicago, Illinois, November 3–5, 1982. IEEE Computer Society Press.

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