Indistinguishability Obfuscation from Constant-Degree Graded Encoding Schemes

Huijia Lin[*]

University of California, Santa Barbara

Abstract

We construct a general-purpose indistinguishability obfuscation (IO) scheme for all polynomial-size circuits from *constant-degree* graded encoding schemes in the plain model, assuming the existence of a subexponentially secure Pseudo-Random Generator (PRG) computable by constant-degree arithmetic circuits (or equivalently in NC^0), and the subexponential hardness of the Learning With Errors (LWE) problems. In contrast, previous general-purpose IO schemes all rely on polynomial-degree graded encodings.

Our general-purpose IO scheme is built upon two key components:

- a new bootstrapping theorem that subexponentially secure IO for a subclass of *constant-degree arithmetic circuits* implies IO for all polynomial size circuits (assuming PRG and LWE as described above), and
- a new construction of IO scheme for any generic class of circuits in the ideal graded encoding model, in which the degree of the graded encodings is bounded by a variant of the degree, called type degree, of the obfuscated circuits.

In comparison, previous bootstrapping theorems start with IO for NC^1, and previous constructions of IO schemes require the degree of graded encodings to grow polynomially in the size of the obfuscated circuits.

[*]`rachel.lin@cs.ucsb.edu`. Huijia Lin was partially supported by NSF grants CNS-1528178 and CNS-1514526. This work was done in part while the author was visiting the Simons Institute for the Theory of Computing, supported by the Simons Foundation and by the DIMACS/Simons Collaboration in Cryptography through NSF grant CNS-1523467.

Contents

1 Introduction **1**
1.1 Overview . . . 2
1.1.1 Part 1: Bootstrapping IO for constant-degree arithmetic circuits . . . 3
1.1.2 Part 2: Special purpose IO in constant-degree ideal graded encoding model . 5
1.1.3 Part 3: Instantiation with Concrete Graded Encoding Schemes . . . 6
1.1.4 Generalization to Arbitrary PRGs . . . 7
1.2 Low Depth PRGs . . . 8
1.3 Organization . . . 8

2 Preliminaries **9**
2.1 Models of Computation . . . 9
2.2 μ-Indistinguishability . . . 11
2.3 Indistinguishability Obfuscation . . . 11
2.4 Indistinguishability Obfuscation with Universal Efficiency . . . 12
2.5 Pseudorandom Generator . . . 13
2.6 Puncturable Pseudo-Random Functions . . . 13
2.7 Functional Encryption . . . 13
2.8 Randomized Encoding . . . 15

3 Bootstrapping IO for Special-Purpose Circuits **15**
3.1 Instantiating the Underlying Primitives from LWE . . . 16
3.2 Overview . . . 16
3.3 FE for $\mathsf{NC}^{1,T,n}$ from IO for $\{\mathcal{P}_\lambda^{T,n}\}$ . . . 17
3.3.1 Proof of Lemma 2 . . . 19
3.3.2 Setting Parameters . . . 22
3.4 Obtaining IO for $\mathsf{P/poly}$ . . . 22

4 Special-Purpose Circuits in Constant Degree **23**
4.1 Overview . . . 23
4.2 Performing Comparisons in Constant-Degree . . . 25
4.3 PRF Evaluation in Constant-Degree . . . 25
4.4 Putting Pieces Together . . . 27

5 Graded Encoding Schemes and Ideal Graded Encoding Oracle **28**
5.1 Graded Encoding Schemes . . . 29
5.2 The Level Function and Level-Respecting Arithmetic Circuits . . . 30
5.3 Ideal Graded Encoding Oracle . . . 31
5.4 IO in Oracle Model . . . 32

6 IO for Special-Purpose Circuits in Ideal Model **32**
6.1 Overview . . . 33
6.2 Type-Degree of Arithmetic Circuits . . . 35
6.3 Type-Degree Preserving Construction of IO . . . 38
6.4 Type Degree Preserving Construction of IO . . . 40
6.5 Efficiency of Our IO Scheme . . . 46
6.6 Unbounded Simulation Security . . . 47

7 IO from Constant-Degree Semantically Secure Graded Encodings **52**
7.1 Distributional Semantic Security of GE schemes 53
7.2 Instantiating Our IO Scheme in the Plain Model 54

8 Proof Sketch of Theorem 4 **61**

1 Introduction

Program obfuscation [BGI$^+$01] aims to make computer programs unintelligible while preserving their functionality. Recently, the first candidate general purpose indistinguishability obfuscation (IO) scheme for all polynomial-size circuits was proposed by Garg et. al. [GGH$^+$13b]. Soon after that, an explosion of follow-up works showed the impressive power of IO, not only in obtaining classical cryptographic primitives, from one-way functions [KMN$^+$14], trapdoor permutations [BPW16], public-key encryption [SW14a], to fully homomorphic encryption [CLTV15], but also in reaching new possibilities, from functional encryption [GGH$^+$13b], 2-round adaptively secure multiparty computation protocols [GP15, DKR15, CGP15], succinct garbling in time independent of the computation time [CHJV15, BGL$^+$15, KLW15, LPST15], to constant-round concurrent ZK protocol [CLP15]. It seems that IO is charting a bigger and more desirable map of cryptography.

However, the Achilles heel of IO research is that it is still unknown whether general purpose IO can be based on standard hardness assumptions. So far, all general purpose IO schemes are constructed in two steps [BR14, BGK$^+$14, PST14a, GLSW14, AGIS14, Zim15, AB15]. First, an IO scheme for (polynomial size) NC^1 circuits is built using some candidate graded encoding schemes. The latter is an algebraic structure, introduced by Garg, Gentry and Halevi [GGH13a], that enables homomorphically evaluating certain polynomials over encoded ring elements and testing whether the output is zero. Next, a bootstrapping theorem transforms an IO scheme for NC^1 into one for $\mathsf{P/poly}$, assuming the LWE assumption [GGH$^+$13b].

Tremendous efforts have been spent on basing the first step on more solid foundations. Unfortunately, the state of affairs is that all candidate graded encoding schemes [GGH13a, CLT13, CLT15, GGH15] are susceptible to the so called "zeroizing attacks" [GGH13a, CHL$^+$15, GHMS14, BWZ14, CGH$^+$15] to different degrees.

In this work, we approach the problem from a different, more complexity theoretic, angle.

> *How much can we strengthen the bootstrapping theorem, and hence, simplify the task of building graded encoding schemes?*

We explore answers to this question and obtain the following main result:

Theorem 1 (Informal)**.** *Assuming constant-degree PRG and LWE with subexponential hardness, there is a general purpose IO scheme using only* constant-degree *graded encodings.*

Though our result does not eliminate the need of graded encoding schemes, it weakens the requirement on them to *only supporting evaluation of constant-degree polynomials*; such a scheme is referred to as a constant-degree graded encoding scheme. In comparison, previous IO schemes rely on graded encodings with degree polynomial in the size of the obfuscated circuit. This improvement is established via a stronger bootstrapping theorem.

- *Bootstrapping IO for constant-degree arithmetic circuits.* We show that there is a class $\mathcal{C}$ of special-purpose circuits, which is a subclass of constant-degree arithmetic circuits (i.e., corresponding to constant-degree polynomials), such that, subexponentially secure special-purpose IO for $\mathcal{C}$ can be bootstrapped to general purpose IO for $\mathsf{P/poly}$, assuming LWE and the existence of a (super-linear stretch) PRG computable by constant-degree arithmetic circuits, referred to as constant-degree PRG, both with subexponential hardness.

- *Constant-degree graded encodings suffice.* Then, we show that special purpose IO for $\mathcal{C}$ can be constructed from constant-degree graded encoding schemes.

On the Assumption of Constant-Degree PRG and Generalization. Our results rely on the assumption that there exists a sub-exponentially secure constant-degree PRG. In our model of constant degree computations (explained shortly below), such a PRG is also in NC^0, by [NS94]. A candidate of such a PRG is Goldreich's PRG [Gol00], which is closely related with systems of random k-CSP and has been studied in the contexts of complexity theory and cryptography in NC^0.

All our results can be generalized to work with arbitrary, not necessarily constant-degree, PRGs. The generalization is parameterized with the degree of the underlying PRG. It shows an interesting connection between the the degree $d(\lambda)$ of PRG, with the degree $d'(\lambda)$ of the circuits that are "bootstrappable" for IO, and the degree $d''(\lambda)$ of the graded encodings sufficient for obtaining general purpose IO — they are all polynomially related. Because of the general connection w.r.t. the degree of PRG, we choose to use the term constant-degree PRG (as opposed to PRG in NC^0) in this paper.

Relation with recent works [Pas15, MMN15, BV15a]. At a first glance, our main theorem is surprising in light of the recent results by [Pas15, MMN15, BV16]. They showed that any general-purpose IO scheme using *ideal* constant-degree graded encodings can be transformed into an IO scheme in the plain model. Alternatively, their results can be interpreted as: Ideal constant-degree graded encodings do not "help" constructing general-purpose IO schemes. In contrast, our results says that *concrete* constant-degree graded encodings imply general-purpose IO (assuming sub-exponentially secure constant-degree PRG and LWE). The divide stems from the fact that ideal graded encodings can only be used in a black-box manner, whereas our IO scheme crucially makes non-black-box use of the underlying graded encoding scheme. Because of the non-black-box nature of our construction, we actually do not obtain an IO scheme for $\mathsf{P/poly}$ in the ideal constant-degree graded encoding model, and hence we cannot apply the transformation of [Pas15, MMN15, BV16] to eliminate the use of graded encodings. Also because of its non-black-box nature, our construction also circumvents the recent lower bound on black-box construction of IO from a variety of cryptographic primitives by [MMN+16].

Moreover, it is interesting to note that our construction of IO for $\mathsf{P/poly}$ uses as a component the transformation from sub-exponentially secure compact functional encryption to general purpose IO by [BV15a, AJ15]. Their transformation makes non-black-box use to the underlying functional encryption, and is in fact the only non-black-box component in our construction. (See Remark 5 for more details.) Therefore, if there were a black-box transformation from sub-exponentially secure compact functional encryption to general purpose IO, we would have obtained a general purpose IO scheme in the ideal constant-degree graded encoding model, and then by [Pas15, MMN15, BV16], a general purpose IO in the plain model. In summary, the following corollary suggests another avenue towards realizing IO.

Corollary 1. *Assume constant-degree PRG and LWE (with subexponential hardness). If there is a black-box transformation from any (subexponentially secure) compact functional encryption to an IO scheme for* $\mathsf{P/poly}$*, there is an IO scheme for* $\mathsf{P/poly}$ *in the plain model.*

1.1 Overview

Our results contain three parts: First, we establish a stronger bootstrapping theorem from IO for a class $\{\mathcal{C}_\lambda\}$ of special-purpose constant-degree arithmetic circuits to general-purpose IO. Second, we show that thanks to the constant-degree property and the simple structure of the special-purpose circuits, IO for $\{\mathcal{C}_\lambda\}$ can be constructed using only constant-degree graded encodings. The construction of the special-purpose IO scheme makes only black-box use of the constant-degree graded encodings, and is secure in the ideal model; but, the bootstrapping requires using the code of the special-purpose IO scheme. Therefore, to stitch the two first parts together, in the third part, we instantiate the special-purpose IO scheme using semantically secure graded encodings (c.f. [PST14a]), and obtain general-purpose IO via bootstrapping. Below, we explain each part in more detail.

1.1.1 Part 1: Bootstrapping IO for constant-degree arithmetic circuits

So far, there are only two bootstrapping techniques in the literature, both starting from IO for NC^1. The first technique, proposed by [GGH+13b], combines fully homomorphic encryption (FHE) and IO for NC^1, where the latter is used to obfuscate a circuit that performs FHE decryption and verifying the correctness of a computation trace, both can be done in logarithmic depth. The second technique by [CLTV15] is based on Applebaum's idea of bootstrapping VBB for NC^0 [App14a], where the underlying IO for NC^1 is used for obfuscating a circuit that computes for each input a randomized encoding (w.r.t. that input and the obfuscated circuit), using independent randomness produced by a Puncturable Pseudo-Random Functions (PPRF) [SW14b] computable in NC^1 [BV15b].

In sum, current bootstrapping techniques require the basic IO scheme to be able to handle complex cryptographic functions. It is an interesting question to ask what is the simplest circuit class — referred to as a "seed class" — such that, IO for it is sufficient for bootstrapping. In this work, we reduce the complexity of "seed classes" from NC^1 to circuits computable in constant degree. More specifically,

Proposition 1 (Informal, bootstrapping constant-degree computations). *Assume constant-degree PRG and LWE with subexponential hardness. There is a class of special-purpose* constant-degree *circuits $\{C_\lambda\}$ with domains $\{\mathbb{D}_\lambda\}$, where $\mathbb{D}_\lambda \subseteq \{0,1\}^{\text{poly}(\lambda)}$, such that, subexponentially secure IO for $\{C_\lambda\}$ with* universal efficiency *can be bootstrapped into IO for P/poly.*

Let us explain our model of computation and efficiency in more detail.

Model of Constant-Degree Computations and Universal Efficiency Every arithmetic circuit AC naturally corresponds to a polynomial by associating the i^{th} input wire with a formal variable x_i; the degree of AC is exactly the degree of the polynomial. In this work, we consider using arithmetic circuit to compute Boolean functions $f : \{0,1\}^n \to \{0,1\}^m$, or logic circuits C. A natural model of computation is the following: Fix a ring $\mathcal{R}$ (say, the integers or the reals); a Boolean function f, or logic circuit C, is computed by an arithmetic circuit AC, if $\forall x \in \{0,1\}^n$, $C(x) = AC(x)$ over $\mathcal{R}$ (the 0 and 1 bits are mapped to the additive and multiplicative identities of $\mathcal{R}$ respectively). In this work, we consider a even weaker computation model that requires AC to agree with C over any choice of ring $\mathcal{R}$.

- *Constant-Degree Computations:* We say that a logic circuit C is computed by an arithmetic circuit AC, if $\forall x \in \{0,1\}^n$, $C(x) = AC(x)$, over any ring $\mathcal{R}$. We say that C has degree d if AC had degree d.

This model of constant-degree computations is quite weak, in fact, so weak that it is equivalent to NC^0. Nisan and Szegedy [NS94] showed that the degree of the polynomial that computes a Boolean function f over the ring of reals is polynomially related with the decision tree complexity of f. Therefore, if f has constant degree in our model, it has constant decision tree complexity, implying that it is in NC^0. In particular, any constant-degree PRG is also in NC^0, as mentioned above.

On the other hand, it is well known that IO for NC^0 can be trivially constructed by searching for canonical representations, which can be done efficiently as every output bit is computed by a constant-size circuit. Though it would be ideal to bootstrap IO for NC^0, we do not achieve this. Instead, we strengthen the above model of computation by considering partial Boolean functions, or logic circuits, defined only over a subset $\mathbb{D} \subseteq \{0,1\}^n$ (i.e., we only care about inputs in $\mathbb{D}$).

- *Constant-Degree Computations with Partial Domains:* We say that a logic circuit C with domain $\mathbb{D} \subseteq \{0,1\}^n$ is computed by an arithmetic circuit AC, if $\forall x \in \mathbb{D}$, $C(x) = AC(x)$, over any ring $\mathcal{R}$.

A concrete constant-degree partial function that is not computable in NC^0 is a variant of the multiplexer function mux that on input (x, e_i), where $x, e_i \in \{0,1\}^n$ and the hamming weight of e_i is 1, outputs x_i. Clearly, the output bit has to depend on all bits of e_i and cannot be computed in NC^0. But, x_i can be computed in degree 2 as the inner product of x and e_i over any ring $\mathcal{R}$. Indeed, our bootstrapping theorem starts with IO for a class of special purpose circuits that are only defined over a sparse, polynomial sized, domain, embedded in an exponential sized domain $\{0,1\}^n$.

Nevertheless, our model of constant degree computations (with potentially partial domains) is still weak. In particular, it is separated from AC^0, since we cannot compute unbounded AND in it. In the body of the paper, we put even more constraints and say that a class of circuits (as opposed to a single circuit) is constant degree only if they have universal circuits of constant degrees; we omit this detail in the introduction.

As a further evidence on how weak our model of constant degree computations are, we show next that there exists even a statistical IO scheme for such circuits.

Trivial, Statistical IO for Constant Degree Computations Let C be a logic circuit with domain $\mathbb{D} \in \{0,1\}^n$ computable by a degree-d arithmetic circuit AC, which corresponds to a degree-d polynomial p. At a high-level, because degree-d polynomials can be learned in poly(n^d) time, we can obfuscate C in the same time with statistical security. More specifically, the degree-d polynomial $p(x)$ can be equivalently represented as a linear function $L(X)$ over $\ell = n^d$ variables, each associated with a degree d monomial over $x_1 \cdots x_n$. To obfuscate C, we simply pick ℓ inputs $x_1, \cdots, x_\ell \in \mathbb{D}$, such that, their corresponding monomial values $X_1, \cdots, X_\ell$ are linearly independent. Now, the obfuscation $\tilde{C}$ of C is simply the set of input output pairs $(x_1, y_1), \cdots, (x_\ell, y_\ell)$ where $y_i = C(x_i)$ over say $GF(2)$.

Given $\tilde{C}$, we can to evaluate C on any input $x \in \mathbb{D}$, since $C(x) = L(x)$ over any ring, in particular $GF(2)$, and the linear function L can be learned from the set of input output pairs using Gaussian elimination. Moreover, it is easy to see that obfuscation of any two functionally equivalent circuits C and C' are identically distributed, as C and C' have the same truth table and their obfuscations simply reveal a part of their truth tables.

The above construction, though achieve statistical security, is however, trivial: The truth table of a degree-d circuit effectively has at most size n^d (by Gaussian elimination), and the above construction simply publishes the effective truth table. As a result, it is not sufficient for our bootstrapping.

Computationally Secure IO for Constant Degree Computations, with Universal Efficiency. Instead, we require IO for constant degree computations with better, non-trivial, efficiency. More specifically,

- *Universal Efficiency:* We say that IO for constant degree circuits has universal efficiency, if its run-time is independent of the degree of the computation. That is, there is a universal polynomial p, such that, for every d, obfuscating a degree-d circuit C takes time $p(1^\lambda, |C|)$ for sufficiently large λ.

In fact, our bootstrapping theorem works even if the efficiency of IO for constant degree circuits grows with the degree, as long as it is bounded by $n^{h(d)}$ for a sufficiently small function h, say, $h(d) = \log\log(d)$. For simplicity, we consider the above universal efficiency.

Can We Construct IO for Constant Degree Computations with Universal Efficiency Directly? Our model of constant-degree computations is quite weak, and the input to the bootstrapping theorem — IO for a subclass of constant-degree computations with universal efficiency — cuts close to the boundary of known feasibility, such as IO for NC^1, and IO for constant degree computations with non-universal efficiency. In this work, we construct such an IO scheme using graded encodings, and leave open the interesting question whether we can construct such an IO scheme directly, without graded encodings.

Techniques. Towards obtaining IO for $\mathsf{P/poly}$, our starting point is two beautiful recent works by Bitansky and Vaikuntanathan [BV15a] and Ananth and Jain [AJ15] showing that sub-exponentially secure (sublinearly) compact FE for NC^1 implies IO for $\mathsf{P/poly}$. Unfortunately, so far, the former is only constructed from IO for NC^1. Thus, our goal is constructing compact FE using IO for the simplest circuits, which when combined with [AJ15, BV15a] bootstraps to IO for $\mathsf{P/poly}$.

The work of Ananth and Jain [AJ15], and another very recent work by the author, Pass, Seth and Telang [LPST16] already explored this direction: They show that a compact FE scheme for NC^1 circuits with single output bit (which can be based on LWE [GKP+13]) can be transformed into a compact FE for all NC^1 circuits with multiple output bits, using IO for circuits (Turing machines in [AJ15]) with only a logarithmic number $c\log\lambda$ of input wires. Note that such circuits have λ^c-sized truth table, and hence can be trivial obfuscated in size λ^c. [LPST16] shows that as long as the obfuscated circuits has size sub-linear in the size of the truth table $\lambda^{\varepsilon c}$ (matching the sub-linear compactness of FE), the transformation goes through — they say such an IO scheme has non-trivial efficiency.

Based on their works, we further simplify the class of circuits we need to build IO for. Our circuit class has not only polynomial-sized domains, but also only constant degree (as opposed to being in NC^1 as in [AJ15, LPST16]). We achieve this using more refined analysis and a number of new techniques. Namely, we build a special-purpose PPRF for polynomial sized domain that is computable in constant degree. Interestingly, the polynomial-sized domain is not of the form $\{0,1\}^{c\log\lambda}$, rather is embedded sparsely in a much larger domain $\mathbb{D} \subset \{0,1\}^{\mathrm{poly}(\lambda)}$. This crucially allows us to circumvent lower bounds on the complexity of normal PPRF. Moreover, we design ways to perform comparisons, such as, testing $=$, $\geq$, $\leq$ relations, between two integers $i, i' \in [\mathrm{poly}(\lambda)]$ in constant degree; here again, we crucially rely on the fact that we can represent the integers in a different way, embedded sparsely in a much larger domain. The embedding of the inputs is why our special-purpose circuits have only partial domain.

1.1.2 Part 2: Special purpose IO in constant-degree ideal graded encoding model

Ideal graded encoding model [GGH13a, BR14, BGK+14, Zim15, AB15] captures generic algebraic attacks over graded encodings: In this model, players have black-box access to a ring, and can only perform certain restricted operations over ring elements, and determine whether a "legal" polynomial (one satisfying all restrictions) evaluates to 0 or not—this is referred to as a "zero-test query".

An important parameter, called the *degree* of the graded encodings [Pas15, MMN15], is the *maximum degree* of (legal) polynomials that can be "zero-tested". Clearly, the lower the degree is, the weaker of the graded encodings are. Consider for instance, when the degree is one, the ideal graded encoding model is equivalent to the generic group model, in which operations are restricted to be linear (i.e., degree 1 polynomials), and when degree is two, ideal graded encodings capture idealized groups with bilinear maps. Both special cases have been extensively studied.

So far, general-purpose IO schemes in ideal models all require *high degree* graded encodings (polynomial in the *size* of the circuit being obfuscated) [BR14, BGK$^+$14, Zim15, AB15]. The dilemma is that such models are so powerful that even general purpose VBB obfuscation is feasible, which is impossible in the plain model [BGI$^+$01]. Two recent works [Pas15, MMN15] initiated the study of *low-degree* ideal graded encodings, showing that when the degree is restricted to a *constant*, general purpose VBB obfuscation becomes infeasible. Therefore, constant-degree ideal graded encoding model is qualitatively weaker than its high-degree counterpart, and is much closer to the plain model. Nevertheless, we show that it is sufficient for building IO for the simple seed class that our bootstrapping theorem starts with.

Proposition 2 (Informal, special-purpose IO in ideal model)**.** *There is a (sub-exponentially secure) IO scheme for the class $\{C_\lambda\}$ of constant-degree special-purpose circuits in Proposition 1, with universally efficiency, in the constant-degree ideal graded encoding model.*

Our special-purpose IO scheme crucially exploits the constant degree property of our seed class, as well as the simple structure, described below, of circuits in the class.

Type-Degree Preserving IO Construction. Our main technique is characterizing a general type of circuits that admit IO schemes with low degree ideal graded encodings. More specifically, we define a new measure, called *type degree*, for arithmetic circuits, which is a quantity no smaller than the actual degree of the circuit, and no larger than the maximum degree of circuits with the same topology (achieved by a circuit with only multiplication gates). We show that if a class of circuits have type degree td, then there is an IO scheme for this class using ideal graded encodings of roughly the same degree $O(td)$; we say that such an IO construction is *type-degree preserving*. Our type-degree preserving IO construction is based on the IO scheme of Applebaum and Brakerski [AB15] in the composite order ideal graded encoding model; we believe that our construction is of independent interests.

Furthermore, thanks to the simplicity of our special purpose circuits in Proposition 1, we can show that they not only have constant degree, but also have constant type degree, leading to Proposition 2.

1.1.3 Part 3: Instantiation with Concrete Graded Encoding Schemes

The final part combine our bootstrapping theorem (Proposition 1) with our special-purpose IO scheme (Proposition 2) to obtain general-purpose IO, for which we must first instantiate the ideal graded encodings with concrete ones, for the bootstrapping theorem makes non-black-box use of the special-purpose IO. Towards this, the technical question is "under what computational hardness assumption over graded encodings can we prove the security of our special-purpose IO scheme in the plain model?"

So far, in the literature, there are two works that answer questions like the above. Pass, Seth and Telang [PST14a] proposed the meta-assumption of *semantic security* over prime order graded encoding schemes, from which the security of a general purpose IO scheme follows via an explicit security reduction. Subsequently, Gentry, Lewko, Sahai and Waters [GLSW14] proposed the Multilinear Subgroup Elimination assumption over composite order graded encoding schemes which improves upon semantic security in terms of simplicity and the number of assumptions in the family (albeit requiring a sub-exponential security loss).

Following [PST14a], we show that our special purpose IO schemes in Proposition 2 can be instantiated with any composite order graded encoding schemes satisfying an analogue of semantic security for composite order rings; importantly, the graded encoding scheme only need to support constant-degree computation. [1] Hence, combining with our bootstrapping theorem from Part 1, we obtain a general purpose IO scheme from constant-degree graded encoding schemes.

Proposition 3 (Informal, special-purpose IO in the plain model)**.** *There is a (subexponentially secure) IO scheme for the class $\{C_\lambda\}$ of constant-degree special-purpose circuits in Proposition 1, with universally efficiency, assuming (subexponentially) semantically-secure constant-degree graded encodings.*

Finally, applying our bootstrapping theorem (Proposition 1) on the special-purpose IO scheme in the above proposition, gives our main theorem (Theorem 1).

We note here that semantic security for graded encodings [PST14a] is a strong assumption, and our variant for composite order is even slightly stronger. Minimizing the assumption on graded encodings is a very important question, but, however, not the focus of this paper. We merely use semantic security to stitch the first two parts of the paper together, which are our main contributions.

1.1.4 Generalization to Arbitrary PRGs

All our techniques and results are not restricted to work with only constant-degree PRGs, and can be parameterized w.r.t. the degree of the underlying PRG. The parametrization actually shows an interesting connection between the degree d of the PRG, with the degree d' of the seed class for IO, and with the degree d'' of the graded encodings sufficient for constructing general purpose IO — they are all polynomially related.

Proposition 4 (Informal, general bootstrapping theorem)**.** *Assume a degree-$d(\lambda)$ PRG scheme* PRG *and LWE with subexponential hardness. There is a class of constant-degree special-purpose* oracle *circuits $\{\mathcal{C}_\lambda\}$, for which the following holds.*

- General Bootstrapping Theorem: *Subexponentially secure IO for $\{\mathcal{C}_\lambda^{\mathsf{PRG}}\}$ with universal efficiency can be bootstrapped into IO for* $\mathsf{P/poly}$*. Moreover, circuits in $\{\mathcal{C}_\lambda^{\mathsf{PRG}}\}$ have degree* $\mathrm{poly}(d(\lambda))$.
- Special-Purpose IO in Ideal Model: *There exists a (subexponentially secure) IO scheme for $\{\mathcal{C}_\lambda^{\mathsf{PRG}}\}$ with universal efficiency in degree-$\mathrm{poly}(d(\lambda))$ ideal graded encoding model.*

Therefore, assuming additionally subexponentially semantically secure degree-$\mathrm{poly}(d(\lambda))$ graded encoding schemes, there is an IO scheme for $\mathsf{P/poly}$ *in the plain model.*

When plugging in a constant-degree PRG, we immediately obtain Proposition 1, 2 and 3. Other interesting special cases include: 1) when plugging in a PRG in AC^0 or TC^0, the general

[1] We note that the security of (variants of) our IO scheme could potentially be proven from the multilinear subgroup elimination assumption of [GLSW14]; we leave this as future work.

bootstrapping theorem says IO for AC^0 or TC^0 with universal efficiency suffices for constructing general purpose IO, 2) when plugging in a PRG with polylogarithmic degree, general purpose IO can be based on graded encodings with polylogarithmic degree.

Technically, the general bootstrapping theorem follows immediately from the fact that the seed class of circuits we construct in Part 1 makes only black-box calls to the underlying PRG. Hence, it can be viewed as a class $\{\mathcal{C}_\lambda\}$ of constant-degree *oracle circuits* (such a circuit can be computed by an arithmetic circuit with additional oracle gates, and its degree is the degree of the arithmetic circuit when replacing the oracle gates with additions). When plugged-in with an arbitrary PRG of degree $d(\lambda)$, we show that the composed circuits, denoted as $\{\mathcal{C}_\lambda^{\mathsf{PRG}}\}$ have degree, as well as type degree, $\text{poly}(d(\lambda))$. Then, applying our type-degree preserving IO construction in Part 2, yields immediately a special-purpose IO for the class of composed circuits using degree-$\text{poly}(d(\lambda))$ ideal graded encodings. Instantiating the ideal graded encodings with concrete semantically secure ones gives the final statement in the above Proposition.

1.2 Low Depth PRGs

We briefly survey constructions of low depth PRGs. (See Applebaum's book [App14b] for more references and discussions.)

The existence of PRG in TC^0 follows from a variety of hardness assumption including intractability of factoring, discrete logarithm, or lattice problems (e.g. [NR95, NR97, NRR00, BPR12]). Literature on PRG in AC^0 is limited; more works focus directly on PRG in NC^0. On the negative side, it was shown that there is no PRG in NC^0_4 (with output locality 4) achieving super-linear stretch [CM01, MST03]. On the positive side, Applebaum, Ishai, and Kushilevitz [AIK04] showed that any PRG in NC^1 can be efficiently "compiled" into a PRG in NC^0 using randomized encodings, but with only *sub-linear* stretch. The authors further constructed a *linear-stretch* PRG in NC^0 under a specific intractability assumption related to the hardness of decoding "sparsely generated" linear codes [AIK08], previously conjectured by Alekhnovich [Ale03]. Unfortunately, to the best of our knowledge, there is no construction of PRG in NC^0 (or even AC^0) with super-linear stretch from well-known assumptions. But, candidate construction exists.

Goldreich's Candidate PRGs in NC^0. Goldreich's one-way functions $f : \{0,1\}^n \to \{0,1\}^m$ where each bit of output is a fixed predicate P of a constant number d of input bits chosen at random, is also a candidate PRG when $m > n$. Several works investigated the (in)security of Goldreich's OWFs and PRGs: So far, there are no successful attacks when the choice of the predicate P avoids certain degenerating cases [CEMT09, BQ12, OW14, AL15]. Notably, O'Donnell and Witmer [OW14] gave evidence for the security of Goldreich's PRGs with super-linear stretch, showing security against both subexponential-time $\mathbb{F}_2$-linear attacks, as well as subexponential-time attacks using SDP hierarchies such as Sherali-Adams$^+$ and Lasserre/Parrilo.

1.3 Organization

We provide more detailed technical overviews at the beginning of Section 3, 4, and 6.

In section 2, we formalize our model of constant-degree computations, IO with universal efficiency, and provide basic preliminaries. Definitions related to graded encoding schemes and ideal graded encoding models are in Section 5. In Section 3, we prove a prelude of our bootstrapping theorem that identifies a class of special purpose circuits, such that IO for this class with universal efficiency can be bootstrapped to general purpose IO. In Section 4, we show that the class of special purpose circuits identified in Section 3 are computable in constant degree, when the underlying

PRG is. Then, we construct a universally efficient IO scheme for these special purpose circuits in constant-degree ideal graded encoding model in Section 6. We show that our special-purpose IO scheme can be instantiated with any constant-degree graded encoding scheme with semantic security in Section 7.

2 Preliminaries

Let $\mathbb{Z}$ and $\mathbb{N}$ denote the set of integers, and positive integers, $[n]$ the set $\{1, 2, \ldots, n\}$, $\mathcal{R}$ denote a ring, and $\mathbf{0}$,$\mathbf{1}$ the additive and multiplicative identities.

We denote by PPT probabilistic polynomial time Turing machines. The term negligible is used for denoting functions that are (asymptotically) smaller than one over any 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 $\nu(n) < n^{-c}$.

2.1 Models of Computation

Logic Circuits and Partial Domains In this work, by circuit, we mean logic circuits from $\{0,1\}^*$ to $\{0,1\}^*$, consisting of input gates, output gates, and logical operator gates (AND and OR gates with fan-in 2 and fan-out > 0, and NEG gate with fan-in 1).

Any circuit with n-bit input wires and m-bit output wires defines a total Boolean function f mapping $\{0,1\}^n$ to $\{0,1\}^m$. In this work, importantly, we also consider partial functions f defined only over a (partial) domain $\mathbb{D} \subset \{0,1\}^n$. Correspondingly, we associate a circuit C with a domain $\mathbb{D} \subset \{0,1\}^n$, meaning that we only care about evaluating C over inputs in $\mathbb{D}$.

Arithmetic Circuits We also consider arithmetic circuits AC consisting of input gates, output gates and operator gates for addition, subtraction, and multiplication (with fan-in 2 and fan-out > 0). Every arithmetic circuit AC with n input gates defines a n-variate polynomial P over $\mathbb{Z}$, by associating the i^{th} input gate with a formal variable x_i. We say that AC has degree d if P has degree d. An arithmetic circuit AC can also be evaluated over any other ring $\mathcal{R}$ (different from $\mathbb{Z}$), corresponding to computing the polynomial P over $\mathcal{R}$.

Boolean Functions Computable by Arithmetic Circuits In this work, we, however, do not consider evaluating arithmetic circuits over any specific ring. Rather, we say that a Boolean function f from domain $\mathbb{D} \subseteq \{0,1\}^n$ to range $\{0,1\}^m$, is computed/implemented by an arithmetic circuit AC if for every input $x \in \mathbb{D}$ with output $y = C(x)$, AC evaluated on $\mathbf{x}$ equals to $\mathbf{y}$ over *any* ring $\mathcal{R}$, where $\mathbf{x}$ and $\mathbf{y}$ are vectors of ring elements derived from x and y respectively, by mapping 0 to the additive identity $\mathbf{0}$ and 1 to the multiplicative identity $\mathbf{1}$ of $\mathcal{R}$. We omit explicitly mentioning this conversion in the rest of the paper, and simply write $AC(x) = C(x)$.

We stress again that, in our model, a Boolean function f is computable by an arithmetic circuit only if it produces the correct outputs for all inputs in $\mathbb{D}$, *no matter what underlying ring is used.* This restriction makes this model of computation very weak.

Similarly, we say that a circuit C with domain $\mathbb{D} \subset \{0,1\}^n$ is computable by an arithmetic circuit AC, if the Boolean function $f : \mathbb{D} \to \{0,1\}^m$ defined by C is computable by AC.

Circuit Classes and Families of Circuit Classes We use the following terminologies and notations:

- A family of circuits $\mathcal{C}$ with domain $\mathbb{D}$ is simply a set of circuits $C \in \mathcal{C}$ with common domain $\mathbb{D}$.
- A class of circuits $\{\mathcal{C}_\lambda\}_{\lambda\in\mathbb{N}}$ with domains $\{\mathbb{D}_\lambda\}_{\lambda\in\mathbb{N}}$ is an ensemble of sets of circuits, where each $\mathcal{C}_\lambda$ is associated with domain $\mathbb{D}_\lambda$. We use the shorthands $\{\mathcal{C}_\lambda\}$ and $\{\mathbb{D}_\lambda\}$.
- A family of circuit classes $\{\{\mathcal{C}^x_\lambda\}\}^{x\in X}$ is a set of circuit classes, where each circuit class $\{\mathcal{C}^x_\lambda\}$ is indexed by an element x in a (countable) index set X. For convenience, when the index set X is clear in the context, we use shorthand $\{\{\mathcal{C}^x_\lambda\}\}$. A family of circuit classes can also be associated with domains $\{\{\mathbb{D}^x_\lambda\}\}$, meaning that each set $\mathcal{C}^x_\lambda$ is associated with domain $\mathbb{D}^x_\lambda$

  For example, NC^1 circuits can be described as a family of circuit classes $\left\{\{\mathcal{C}^d_\lambda\}\right\}^{d\in\mathbb{N}}$, where for every $d \in \mathbb{N}$, the circuit class $\left\{\mathcal{C}^d_\lambda\right\}$ contains all circuits of depth $d \log \lambda$.

Universal (Arithmetic) Circuits Let $\mathcal{C}$ be a family of circuits with domain $\mathbb{D}$, where every $C \in \mathcal{C}$ is described as an ℓ-bit string, and let $\mathcal{U}$ be an (arithmetic) circuit. We say that $\mathcal{U}$ is the universal (arithmetic) circuit of $\mathcal{C}$ if every $C \in \mathcal{C}$ is computed by $\mathcal{U}(\star, C)$ over domain $\mathbb{D}$. Moreover, we say that an ensemble of (arithmetic) circuits $\{\mathcal{U}_\lambda\}$ is the universal (arithmetic) circuits of a circuit class $\{\mathcal{C}_\lambda\}$ with domain $\{\mathbb{D}_\lambda\}$ if for every λ, $\mathcal{U}$ is an (arithmetic) universal circuit of $\mathcal{C}_\lambda$ with domain $\mathbb{D}_\lambda$.

Degree of (Logic) Circuits Degree is naturally defined for arithmetic circuits as described above, but not so for logic circuits and Boolean functions. In this work, we define the degrees of logic circuits and Boolean functions through the degree of the arithmetic circuits that compute them. Moreover, we also define degrees for families of circuits, circuit classes, and families of circuit classes, through the degrees of the universal arithmetic circuits that compute them.

Degree of a (logic) circuit: We say that a circuit C with domain $\mathbb{D}$ has degree d, if it is computable by an arithmetic circuit of degree d.

Degree of a family of circuits: We say that a family of circuits $\mathcal{C}$ with domain $\mathbb{D}$ has degree d, if it has a universal arithmetic circuit $\mathcal{U}$ of degree d.

Degree of a class of circuits: We say that a class of circuits $\{\mathcal{C}_\lambda\}$ with domain $\mathbb{D}_\lambda$ has degree $d(\lambda)$, if it has universal arithmetic circuits $\{\mathcal{U}_\lambda\}$, with degree $d(\lambda)$. If $d(\lambda)$ is a constant function, then we say $\{\mathcal{C}_\lambda\}$ has constant degree.

Degree of a family of circuit classes: We say that a family of circuit classes $\{\{\mathcal{C}^x_\lambda\}\}$ with domains $\{\{\mathbb{D}^x_\lambda\}\}$ has constant degree, if for every $x \in X$, circuit class $\{\mathcal{C}^x_\lambda\}$ with domains $\{\mathbb{D}^x_\lambda\}$ has constant degree.

It is important to note that we define the degree of a class of circuits via the degree of its *universal* arithmetic circuit, not the degree of individual circuits inside. For example, consider the natural class of circuits containing all (polynomial-sized) circuits with a fixed constant degree d (c.f., the class of poly-sized NC^0 circuits with a fixed constant depth d), under our definition, it is not clear whether this class itself has constant degree, as it is not clear (to us) whether there is a constant degree universal arithmetic circuit that computes all of them. Nevertheless, this more stringent definition only makes our bootstrapping result that it suffices to construct IO for a family of circuit classes with constant degree stronger, and makes the task of constructing IO for such a family easier.

The Nisan-Szegedy Result. Nisan and Szegedy [NS94] studied the relation between a total Boolean function $f : \{0,1\}^n \to \{0,1\}$, and the polynomial p that computes it over *reals*. They proved that the degree d of p is polynomially related with the decision tree $D(f)$ complexity of f, in particular, $d \leq D(f) \leq 16d^8$. Therefore, if d is a constant, $D(f)$ is also a constant, and hence the output bit relies on at most $D(f)$ number of input bits.

Under our model, a function is computed by an arithmetic circuit only if the latter agrees with the former over any ring, in particular, the reals. Thus, we have,

Corollary 2 ([NS94])**.** *If a* total *Boolean function $f : \{0,1\}^n \to \{0,1\}$ can be computed by an arithmetic circuit of degree d (in our model), its output bit depends on at most $16d^8$ input bits.*

In contrast, if a Boolean function f is only defined over a partial domain $\mathbb{D} \subset \{0,1\}^n$, then the above is not true. Roughly speaking, this Corollary implies that any constant-degree PRG in our model is also in NC^0.

2.2 μ-Indistinguishability

Definition 1 (μ-indistinguishability)**.** *Let $\mu : \mathbb{N} \to [0,1]$ be a function. A pair of distribution ensembles $\{X_\lambda\}$, $\{Y_\lambda\}$ are μ-indistinguishable if for every non-uniform PPT distinguisher D, every sufficiently large security parameter $\lambda \in \mathbb{N}$, and auxiliary input $z \in \{0,1\}^{\mathrm{poly}(\lambda)}$, it holds that*

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

Definition 2 (Computational and Sub-exponential Indistinguishability)**.** *A pair of distribution ensembles $\{X_\lambda\}$, $\{Y_\lambda\}$ 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 distinguisher running in sub-exponential time.

Below, we provide definitions of standard cryptographic primitives using μ-indistinguishability, which implicitly define variants with polynomial or sub-exponential security.

2.3 Indistinguishability Obfuscation

We recall the notion of indistinguishability obfuscation for a class of circuit defined by [BGI⁺01], adding the new dimension that the class of circuits may have restricted domains $\{\mathbb{D}_\lambda\}$.

Definition 3 (Indistinguishability Obfuscator ($i\mathcal{O}$) for a circuit class)**.** *A uniform* PPT *machine $i\mathcal{O}$ is a indistinguishability obfuscator for a class of circuits $\{\mathcal{C}_\lambda\}_{\lambda \in \mathbb{N}}$ (with potentially restricted domains $\{\mathbb{D}_\lambda\}_{\lambda \in \mathbb{N}}$), 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 (in $\mathbb{D}_\lambda$), we have that*

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

μ-Indistinguishability: *For every ensemble of pairs of circuits $\{C_{0,\lambda}, C_{1,\lambda}\}_\lambda$ 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 (in $\mathbb{D}_\lambda$), the following ensembles of distributions are μ-indistinguishable,*

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

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

Definition 4 (IO for $\mathsf{P/poly}$). *A uniform* PPT *machine* $i\mathcal{O}_{\mathsf{P/poly}}(\star,\star)$ *is an indistinguishability obfuscator for* $\mathsf{P/poly}$ *if it is an indistinguishability obfuscator for the class* $\{\mathcal{C}_\lambda\}$ *of circuits of size at most* λ*.*

2.4 Indistinguishability Obfuscation with Universal Efficiency

In this work, we consider families of circuit classes, and the task of building a family of indistinguishability obfuscators, one for each circuit class.

Definition 5 (IO for Families of Circuit Classes). *Let* $\{\{\mathcal{C}^x_\lambda\}\}^{x\in X}$ *be a family of circuit classes (with potentially restricted domains* $\{\mathbb{D}^x_\lambda\}$*). A family of uniform machines* $\{i\mathcal{O}^x\}^{x\in X}$ *is a family of indistinguishability obfuscators for* $\{\{\mathcal{C}^x_\lambda\}\}^{x\in X}$*, if for every constant* $x\in X$*,* $i\mathcal{O}^x$ *is an indistinguishability obfuscator for the circuit class* $\{\mathcal{C}^x_\lambda\}$ *(with domains* $\{\mathbb{D}^x_\lambda\}$*).*

The above definition implicitly requires that for every $x\in X$, $i\mathcal{O}^x$ runs in some polynomial time, potentially depending on x. However, depending on how the run-time of $i\mathcal{O}^x$ vary for different x, qualitatively different types of efficiency could be considered.

Universal Efficiency: In one end, the run-time of $i\mathcal{O}^x$ could be (almost) independent of x.

For example, in [GGH+13b], IO for NC^1 is defined as a *uniform* PPT machine $i\mathcal{O}_{\mathsf{NC}^1}(\star,\star,\star)$, such that, for every $c\in\mathbb{N}$, $i\mathcal{O}^c_{\mathsf{NC}^1}(\star,\star) = i\mathcal{O}_{\mathsf{NC}^1}(c,\star,\star)$ is an indistinguishability obfuscator for the class of circuits of depth at most $c\log\lambda$ and size at most λ.

Given that $i\mathcal{O}_{\mathsf{NC}^1}$ is uniform PPT, the run-time of $i\mathcal{O}^c_{\mathsf{NC}^1}$ is essentially independent of c. In particular, there exists a universal polynomial p, such that, for every index c, the run-time of $i\mathcal{O}^c_{\mathsf{NC}^1}$ is bounded by $p(\lambda,|C|)$ for every sufficiently large λ (say $>|c|$).

We refer to this type of efficiency *universal efficiency*, as the run-time of different $i\mathcal{O}^x$'s are bounded by a universal polynomial.

Non-Universal Efficiency: In the other end, the run-time of $i\mathcal{O}^x$ could depend (heavily) on x.

For example, the work of [GGH+13b] (and many other follow-up works) presents a construction of IO for NC^1 that runs in time $\mathrm{poly}(\lambda,|C|)^c$ (as opposed to a universal polynomial independent of c). This is because their construction uses branching programs, which has size $O(1)^{c\log\lambda}$ for circuits with depth $c\log\lambda$.

To differentiate, we call this type of efficiency *non-universal efficiency*, as the run-time of different $i\mathcal{O}^x$'s are only bounded by different polynomials p_x.

In the context of IO for NC^1, non-universal efficiency suffices for the purpose of bootstrapping to IO for $\mathsf{P/poly}$ [GGH+13b]. However, as discussed in the introduction, in the context of IO for families of circuit classes with constant degree, non-universal efficiency is trivial to achieve and is insufficient for bootstrapping to IO for $\mathsf{P/poly}$. Therefore, we require universal efficiency, formalized below.

Definition 6 (Universal Efficiency). *A family of indistinguishability obfuscators* $\{i\mathcal{O}^x\}^{x\in X}$ *for a family of circuit class* $\{\{\mathcal{C}^x_\lambda\}\}^{x\in X}$ *(with potentially restricted domains* $\{\{\mathbb{D}^x_\lambda\}\}$*) has* universal efficiency*, if there exists a universal polynomial* p*, such that, for every* $x\in X$*,* $i\mathcal{O}^x(1^\lambda,C)$ *runs in time* $p(\lambda,|C|)$*, for every sufficiently large* λ *(i.e., greater than a constant* c^x *depending on* x*), and circuit* $C\in\mathcal{C}^x_\lambda$*.*

We note that it is without loss of generality to only consider the run-time of $i\mathcal{O}^x$ for sufficiently large λ ($>c^x$), because the security of $i\mathcal{O}^x$ already only holds for sufficiently large λ.

2.5 Pseudorandom Generator

Definition 7 (Pseudo-Random Generator (PRG))**.** *Let ℓ be a computable polynomial. A deterministic polynomial-time uniform machine* PRG *is a $\ell(\lambda)$-stretch pseudorandom generator if the following conditions are satisfied:*

Syntax *For every $\lambda \in \mathbb{N}$ and every $r \in \{0,1\}^\lambda$, $\mathsf{PRG}(r)$ outputs $r' \in \{0,1\}^{\ell(\lambda)}$*

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

$$\left\{r \xleftarrow{\$} \{0,1\}^\lambda : \mathsf{PRG}(r)\right\} \ \left\{r' \xleftarrow{\$} \{0,1\}^{\ell(\lambda)}\right\}$$

For every $\lambda \in \mathbb{N}$, let $\mathsf{PRG}_\lambda : \{0,1\}^\lambda \to \{0,1\}^{\ell(\lambda)}$ denote the total Boolean function corresponding to PRG *for λ bit inputs. we say that* PRG *has degree d, if for every λ,* PRG_λ *has degree d.*

Since every PRG_λ is a total function, by the Nisan-Szegedy result (Corollary 2), PRG must belong to NC^0.

Claim 1. *Every constant-degree* PRG *is in* NC^0.

2.6 Puncturable Pseudo-Random Functions

We recall the definition of puncturable pseudo-random functions (PRF) from [SW14b]. Since in this work, we only uses puncturing at one point, the definition below is restricted to puncturing only at one point instead of at a polynomially many points.

Definition 8 (Puncturable PRFs)**.** *Let n be a computable polynomial and $\left\{\mathbb{D}_\lambda \subseteq \{0,1\}^{n(\lambda)}\right\}$ an ensemble of sets. A puncturable family of PRFs with domains $\{\mathbb{D}_\lambda\}$ 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 $i \in \mathbb{D}_\lambda$, and $K(-i) = \mathsf{PRF.Punc}(K, i)$, we have that $\mathsf{F}(K(-i), x) = \mathsf{F}(K, x)$ for all $x \neq i$.*

Pseudorandom at punctured point: *For every ensemble $\{i_\lambda \in \mathbb{D}_\lambda\}$, 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 pseudorandom generators (PRGs) yields puncturable PRFs. Furthermore, it is easy to see that if the PRG underlying the GGM construction is sub-exponentially hard, then the resulting puncturable PRF is sub-exponentially pseudo-random.

2.7 Functional Encryption

We provide the definition of a public key single query functional encryption (FE) scheme with selective indistinguishability-based security based on that in [BSW12, O'N10].

Definition 9 (Selectively-secure Single-Query Public-key Functional Encryption)**.** *A tuple of algorithms* $(\mathsf{FE.Setup}, \mathsf{FE.KeyGen}, \mathsf{FE.Enc}, \mathsf{FE.Dec})$ *is a selectively-secure functional encryption scheme for a class of circuits* $\{\mathcal{C}_\lambda\}$ *if it satisfies the following properties.*

Completeness: *For every* $\lambda \in \mathbb{N}$*,* $C \in \mathcal{C}_\lambda$*, message* $m \in \{0,1\}^*$*, and* $T > \max(|C|, |m|)$*,*

$$\Pr\left[\begin{array}{c} (mpk, msk) \leftarrow \mathsf{FE.Setup}(1^\lambda, T) \\ c \leftarrow \mathsf{FE.Enc}(1^\lambda, mpk, m) \\ sk_C \leftarrow \mathsf{FE.KeyGen}(msk, C) \end{array} : C(m) \leftarrow \mathsf{FE.Dec}(sk_C, c)\right] = 1$$

Non-Compact Efficiency: *There is a universal polynomials* p*, such that, for every polynomial* T*, there is a constant* c^T *and the following holds:*

For every $\lambda > c^T$*, the above three algorithms* $\mathsf{FE.Setup}$*,* $\mathsf{FE.Enc}$*,* $\mathsf{FE.KeyGen}$ *all run in time* $p(\lambda, T(\lambda))$*, when* $\mathsf{FE.Enc}$ *and* $\mathsf{FE.KeyGen}$ *are fed with correctly generated* mpk*,* msk*, and* C *(as described above in completeness).*

μ**-Selective-security:** *For every polynomial* T*, and every ensemble of circuits and pair of messages* $\{C_\lambda, m_{0,\lambda}, m_{1,\lambda}\}$*, where* $C_\lambda \in \mathcal{C}_\lambda$*,* $|m_{0,\lambda}| = |m_{1,\lambda}| < |C_\lambda| \le T(\lambda)$*, and* $C_\lambda(m_{0,\lambda}) = C_\lambda(m_{1,\lambda})$*, the following ensembles of distributions* $\{D_{0,\lambda}\}$ *and* $\{D_{1,\lambda}\}$ *are* μ*-indistinguishable.*

$$D_{b,\lambda} = \left(\begin{array}{c} (mpk, msk) \leftarrow \mathsf{FE.Setup}(1^\lambda, T(\lambda)) \\ c \leftarrow \mathsf{FE.Enc}(1^\lambda, mpk, m_{b,\lambda}) \\ sk_C \leftarrow \mathsf{FE.KeyGen}(msk, C_\lambda) \end{array} : mpk, sk_C, c\right)$$

We note that in this work, we only need functional encryption supporting a single secret-key query for functions with *multi-bit outputs*. Moreover, the security of the functional encryption scheme only need to hold with respect to statically chosen challenge messages and functions.

Compact Functional Encryption. Non-compact efficiency only requires algorithms of the FE scheme to run in time polynomial in the worst-case run-time of the computation. A stronger notion of efficiency, called compactness (or weakly compactness) was proposed in [AJ15, BV15a], which requires the encryption algorithm to run in time $\text{poly}(\lambda, |m|, \log T)$ (or $\text{poly}(\lambda, |m|)T^{1-\varepsilon}$); in essence, encryption is faster than the computation itself for any sufficiently large polynomial T.

Definition 10 (Compact Functional Encryption [AJ15, BV15a])**.** *We say that a FE scheme for a class of circuits* $\{\mathcal{C}_\lambda\}$ *is* compact *or* $(1-\varepsilon)$ weakly-compact *for a constant* $\varepsilon \in (0,1)$*, if the efficiency requirement of Definition 9 is strengthened to the following:*

Compactness: *The encryption algorithm* $\mathsf{FE.Enc}$ *runs in time*

$$\mathsf{Time}_{\mathsf{FE.Enc}}(1^\lambda, mpk, m) \le p(\lambda, |m|, \log T) \ .$$

$(1-\varepsilon)$**-Weakly Compactness:** *The encryption algorithm* $\mathsf{FE.Enc}$ *runs in time*

$$\mathsf{Time}_{\mathsf{FE.Enc}}(1^\lambda, mpk, m) \le p(\lambda, |m|)T^{1-\varepsilon} \ .$$

2.8 Randomized Encoding

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

Definition 11 (Randomized Encoding Scheme for Circuits)**.** *A Randomized Encoding scheme* RE *consists of two PPT algorithms,*

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

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

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

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

$$\left\{\hat{C}_x \xleftarrow{\$} \mathsf{RE.Enc}(1^\lambda, C, x) : \hat{C}_x\right\}$$
$$\left\{\hat{C}_x \xleftarrow{\$} \mathsf{RE.Sim}(1^\lambda, C(x), 1^{|C|}, 1^{|x|}) : \hat{C}_x\right\}$$

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

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

3 Bootstrapping IO for Special-Purpose Circuits

In this section, we identify a family of special-purpose circuit classes and show how to bootstrap IO for this family to all polynomial-sized circuits.

Proposition 5. *Assume the following primitives:*

- *a sub-exponentially secure compact FE scheme* FE *for Boolean* NC^1 *circuits,*
- *a sub-exponentially secure PPRF scheme* PPRF*, and*
- *a sub-exponentially secure RE scheme* RE *in* NC^0*.*

Then, there is a family of special-purpose circuit classes $\{\{\mathcal{P}_\lambda^{T,n}\}\}$ *indexed by two polynomials* $T(\star)$ *and* $n(\star)$ *and defined w.r.t.* FE*,* PPRF *and* RE *as in Figure 1, such that, the following holds:*

- *If there exists a family* $\{i\mathcal{O}^{T,n}\}$ *of sub-exponentially secure IO schemes for* $\{\{\mathcal{P}_\lambda^{T,n}\}\}$ *with* universal efficiency*, then there are two sufficiently large polynomials* T^* *and* n^**, such that,* $i\mathcal{O}^{T^*,n^*}$ *can be transformed into a (sub-exponentially secure) IO scheme for* $\mathsf{P/poly}$*.*

We note in Section 3.1 that all the underlying primitives of the above Proposition are implied by the sub-exp hardness of LWE, and provide an overview on the proof of the proposition in Section 3.2.

3.1 Instantiating the Underlying Primitives from LWE

The first primitive of Proposition 5—a compact FE for Boolean NC^1 circuits—can be derived from the work of Goldwasser, Kalai, Popa, Vaikuntanathan and Zeldovich [GKP+13]: Assuming sub-exp LWE, they construct a sub-exp secure FE scheme for the class of polynomial-sized *Boolean* circuits $\{\mathcal{C}_{n,d(n)}\}$ with n input bits and depth $d(n)$. Furthermore, the size of the ciphertexts is $\text{poly}(\lambda, n, d)$ (independent of the size of the circuits); when restricting to Boolean circuits in NC^1 (as needed for Proposition 5), the ciphertexts are compact. Summarizing,

Theorem 2 (Compact FE scheme for Boolean NC^1 Circuits [GKP+13])**.** *Assume sub-exponential hardness of the LWE problem. There exists a sub-exponentially secure compact (single-query, public-key) FE scheme for the class of* Boolean NC^1 *circuits.*

The second primitive—a sub-exp secure PPRF—can be constructed from the necessary assumption of sub-exp secure OWFs [BW13, BGI14, KPTZ13]; but, the evaluation algorithms of these PPRF schemes have high depth. Recently, Brakerski and Vaikuntanathan [BV15c] showed that assuming LWE, the depth of the evaluation algorithm can be reduced to logarithmic $O(\log \lambda)$.

Finally, the third primitive—a sub-exp secure RE scheme in NC^0—can be constructed from sub-exp secure low-depth PRG [AIK04, IK02], which is in turn implied by sub-exp secure LWE.

3.2 Overview

Towards the proposition, recall that recent works [BV15a, AJ15, ABSV15] show that to construct IO for $\mathsf{P/poly}$, it suffices to construct a compact FE scheme for NC^1 circuits. Formally,

Theorem 3 ([BV15a, AJ15, ABSV15])**.** *Let n be a sufficiently large polynomial. Assume the existence of a sub-exponentially secure, and $(1-\varepsilon)$-weakly-compact (single-query, public-key) FE scheme for NC^1 circuits, and weak PRF in NC^1. There exists an indistinguishability obfuscator for* $\mathsf{P/poly}$.

Therefore, the natural direction is constructing compact FE scheme for NC^1 circuits using IO for the special-purpose circuits. We proceed in two steps: For any polynomials T and n, let $\mathsf{NC}^{1,T,n}$ be the subclass of NC^1 circuits with at most size $T(\lambda)$ and at most $n(\lambda)$ input bits.

- Our first step (in Section 3.3) constructs a (sub-exponentially secure) FE scheme $\mathsf{FE}^{T,n}$ for $\mathsf{NC}^{1,T,n}$ from any (sub-exponentially secure) IO scheme $i\mathcal{O}^{T,n}$ for $\{\mathcal{P}_\lambda^{T,n}\}$ (and the underlying primitives of Proposition 5), for arbitrary T and n. Importantly, the encryption time of the resulting FE scheme is directly proportional to the obfuscation time of the underlying IO scheme:

$$\begin{aligned}\mathsf{Time}_{i\mathcal{O}^{T,n}}(1^\lambda, C) &\leq p^{T,n}(\lambda, |C|)\\ \mathsf{Time}_{\mathsf{FE.Enc}}(mpk, m) &\leq p^{T,n}(\lambda,\ q(\lambda, n(\lambda), \log T(\lambda)))\end{aligned}$$

  where q is a universal polynomial independent of T and n. Note that, this does not guarantee that the resulting FE scheme is compact, since the run-time of the IO scheme may depend on T arbitrarily, in particular, it is possible that $p^{T,n}(\lambda, |C|) > T(\lambda)$, while $i\mathcal{O}^{T,n}$ is still a valid polynomial time IO scheme for $\{\mathcal{P}_\lambda^{T,n}\}$.

- To overcome the above issue, our next step (in Section 3.4) starts with a stronger premise: The existence of a family $\{i\mathcal{O}^{T,n}\}$ of IO schemes for the family $\{\{\mathcal{P}_\lambda^{T,n}\}\}$ with universal efficiency.

This means for any T, n, the obfuscation time of $i\mathcal{O}^{T,n}$ is bounded by a universal polynomial p, and (for sufficiently large λ)

$$\begin{aligned}\mathsf{Time}_{i\mathcal{O}^{T,n}}(1^\lambda, C) &\leq p(\lambda, |C|)\\ \mathsf{Time}_{\mathsf{FE.Enc}}(mpk, m) &\leq p(\lambda,\ q(\lambda, n(\lambda), \log T(\lambda)))\end{aligned}$$

This essentially means that the FE schemes are compact — encryption time is independent of $T(\lambda)$. In particular, for some sufficiently large polynomials T^* and n^*, encryption time of FE^{T^*,n^*} is much smaller than the time of the computation, that is, $p(\lambda,\ q(\lambda, n^*(\lambda), \log T^*(\lambda))) < T^*$. With a closer examination, such an FE scheme FE^{T^*,n^*} is sufficient for the transformation of [BV15a, AJ15, ABSV15] to go through. More specifically, the final IO scheme for $\mathsf{P/poly}$ they construct only need to use the underlying FE scheme for NC^1 circuits with some sufficiently large size T^* and sufficiently long input length n^*; the proof goes through, as long as encryption time is sub-linearly $(T^*)^{1-\varepsilon}$ in T^*.

Putting the two steps together, we conclude Proposition 5.

Technically, the transformation in the first step is similar to that in [AJ15, LPST16]. However, the former [AJ15] requires IO for a class of special-purpose Turing machines (as opposed circuits). Our transformation uses the same idea as in [LPST16], but requires a much more refined analysis in order to identify and simplify the circuits, whose special structure plays a key role later.

3.3 FE for $\mathsf{NC}^{1,T,n}$ from IO for $\{\mathcal{P}_\lambda^{T,n}\}$

Fix arbitrary polynomials T and n. We present an FE scheme $\mathsf{FE}^{T,n}$ for $\mathsf{NC}^{1,T,n}$ from IO for $\{\mathcal{P}_\lambda^{T,n}\}$. Our construction starts with a compact FE scheme for Boolean NC^1 circuits $\mathsf{bFE} = (\mathsf{bFE.Setup}, \mathsf{bFE.Enc}, \mathsf{bFE.Dec})$ (as discussed in 3.1, such a scheme can be constructed from LWE), and transforms it into $\mathsf{FE}^{T,n}$. The transformation makes uses of the following additional building blocks:

- a puncturable PRF $\mathsf{PPRF} = (\mathsf{PRF.Gen}, \mathsf{PRF.Punc}, \mathsf{F})$ for input domain $\{0,1\}^\lambda$.
- a randomized encoding scheme $\mathsf{RE} = (\mathsf{RE.Enc}, \mathsf{RE.Eval})$ in NC^0, and
- an IO scheme $i\mathcal{O}^{T,n}$ for circuit class $\{\mathcal{P}_\lambda^{T,n}\}$ consisting all circuits of the form $P[\lambda, T, n, mpk, i^*, K, m_1, y, m_0]$ defined in Figure 1.

Let $\ell_{mpk}(\lambda)$ be the maximal length of master public keys of bFE, and $\ell_{key}(\lambda)$ that of punctured keys of PPRF respectively.

Construction of $\mathsf{FE}^{T,n}$. For any λ, $T = T(\lambda)$ and $n = n(\lambda)$, message m of length n and circuit C with size at most T and input length at most n. The FE scheme $\mathsf{FE}^{T,n} = (\mathsf{FE.Setup}, \mathsf{FE.KeyGen}, \mathsf{FE.Enc}, \mathsf{FE.Dec})$ proceeds as follow:

Setup $\mathsf{FE.Setup}(1^\lambda, T)$**:** Samples $(mpk, msk) \xleftarrow{\$} \mathsf{bFE.Setup}(1^\lambda, T')$, where T' is a time bound for circuits $\bar{C}$ defined below.

Key Generation $\mathsf{FE.KeyGen}(msk, C)$**:** Let $\bar{C}(m, i)$ be a circuit that on input m and $i \in [T]$ outputs the ith bit y_i of the output $y = C(m)$.

Sample $sk_{\bar{C}} \leftarrow \mathsf{bFE.KeyGen}(msk, \bar{C})$; output $sk = sk_{\bar{C}}$.

Circuit $P[T, n, mpk, i^*, K, m_<, \hat{\Pi}, m_>](i)$

Constant: A security parameter $\lambda \in \mathbb{N}$, a time bound $T \in \mathbb{N}$, a threshold $i^* \in \{0, \cdots, T+1\}$, a public key $mpk \in \{0,1\}^{\ell_{mpk}}$ of bFE, a punctured key $K \in \{0,1\}^{\ell_{key}}$ of PPRF, strings $m_<$, $m_>$ of equal length n, and an RE encoding $\hat{\Pi}$.

Input: An index $i \in [T]$.

Procedure:

1. $(R_i || R'_i) = \mathsf{F}(K, i)$;
2. If $i < i^*$, set $\hat{\Pi}_i = \mathsf{RE.Enc}\left(1^\lambda,\ \mathsf{bFE.Enc},\ (mpk,\ m_< || i\ ;\ R_i)\ ;\ R'_i\right)$.
3. If $i = i^*$, set $\hat{\Pi}_i = \hat{\Pi}$.
4. If $i > i^*$, set $\hat{\Pi}_i = \mathsf{RE.Enc}\left(1^\lambda,\ \mathsf{bFE.Enc},\ (mpk,\ m_> || i\ ;\ R_i)\ ;\ R'_i\right)$.

Output: Encoding $\hat{\Pi}_i$.

Padding: The hardwired encoding $\hat{\Pi}$ is padded to be of length $\eta'(\lambda, n, \log T)$, and the circuit is padded to be of size $\eta(\lambda, n, \log T)$, for some polynomials η' and η specified in Section 3.3.2.

Circuit classes $\{\mathcal{P}^{T,n}_\lambda\}$ contains all circuits of form $P[\lambda, T(\lambda), n(\lambda), *, *, *, *, *]$, where all wild-card values satisfy length constraints specified above.

Figure 1: Special-Purpose Circuit P

Encryption $\mathsf{FE.Enc}(mpk, m)$**:**

1. Sample $K \xleftarrow{\$} \mathsf{PRF.Gen}(1^\lambda)$, and puncture it at input 0, $K(-0) = \mathsf{PRF.Punc}(K, 0)$.
2. Sample $\tilde{P} \xleftarrow{\$} i\mathcal{O}^{T,n}(1^\lambda, P)$, where $P = P[\lambda, T, n, mpk, 0, K(-0), 0^\lambda, 0^\kappa, m]$ as defined in Figure 1.
3. Output ciphertext $\xi = \tilde{P}$.

Decryption $\mathsf{FE.Dec}(sk, \xi)$**:**

1. Parse ξ as an obfuscated program $\tilde{P}$; for $i \in [T]$, compute $\hat{\Pi}_i = \tilde{P}(i)$.
2. For every $i \in [T]$, decode $c_i = \mathsf{RE.Eval}(\hat{\Pi}_i)$.
3. For $i \in [T]$, evaluate c_i with sk to obtain $y_i = \mathsf{bFE.Dec}(sk, c_i)$.
4. Output $y = y_1 || \cdots || y_T$.

It is clear that all algorithms above are PPT. Below, we first analyze the encryption efficiency of $\mathsf{FE}^{T,n}$ in Lemma 1 and then show its correctness and security in Lemma 2. How large the special purpose circuits in the construction need to be padded to, i.e., the polynomials η and η' in Figure 1, follows implicitly from the security analysis. We thus set η and η' after the security proof in Section 3.3.2.

Lemma 1. *There exists a universal polynomial q, such that,*

$$\begin{aligned} \textit{if} \qquad & \mathsf{Time}_{i\mathcal{O}^{T,n}}(1^\lambda, C) \le p^{T,n}(\lambda, |C|), \\ \textit{then,} \quad & \mathsf{Time}_{\mathsf{FE.Enc}}(mpk, m) \le p^{T,n}(\lambda, q(\lambda, n(\lambda), \log T(\lambda))) \end{aligned}$$

Proof. Towards this, we analyze the efficiency of each step of $\mathsf{FE.Enc}(mpk, m)$:

- It follows from the efficiency of PPRF that Step 1 of $\mathsf{FE.Enc}$ takes a fixed, universal, polynomial time $q_1(\lambda)$.
- It follows from the compactness of bFE that the size of the special purpose circuit P is bounded by and padded to a fixed, universal, polynomial $\eta(\lambda, n, \log T)$ (in Figure 1).
- It follows from the efficiency of $i\mathcal{O}^{T,n}$ that the second step of encryption takes time $\mathsf{Time}_{i\mathcal{O}^{T,n}}(1^\lambda, P) = p^{T,n}(\lambda, \eta(\lambda, n, \log T))$.

Therefore, there exists a sufficiently large universal polynomial q w.r.t. which the lemma holds. $\square$

Lemma 2. *Let* bFE, PPRF, RE, $i\mathcal{O}^{T,n}$ *be defined as above.* FE *is correct and selectively secure for* NC^1 *circuits with* $n(\lambda)$*-bit inputs. Moreover, if all primitives are sub-exponentially secure, so is* FE.

3.3.1 Proof of Lemma 2

Correctness follows from that of bFE, $i\mathcal{O}^{T,n}$, and RE. More precisely, recall that an honestly generated ciphertext $c = \tilde{P}$ of a message m is the obfuscation of the circuit $P = P[\lambda, T, n, mpk, 0, K(-0), 0^\lambda, \hat{\Pi}, m]$, and an honestly generated sk is the secret key $sk_{\bar{C}}$ of bFE for circuit $\bar{C}$. Then, consider the decryption procedure $\mathsf{FE.Dec}(sk, \xi)$:

- In Step 1, by the correctness of $i\mathcal{O}^{T,n}$ and construction of P, evaluating $\tilde{P}$ on input $i \in [T]$ yields an encoding

$$\hat{\Pi}_i = \mathsf{RE.Enc}\left(1^\lambda,\ \mathsf{bFE.Enc},\ (mpk,\ m||i\ ;\ R_i)\ ;\ R'_i\right)$$

- In Step 2, by the correctness of RE, $\hat{\Pi}_i$ decodes to a ciphertext

$$c_i = \mathsf{bFE.Enc}(mpk,\ m||i\ ;\ R_i)$$

- In Step 3, by the correctness of bFE, evaluating c_i with $sk = sk_{\bar{C}}$ yields $y_i = \bar{C}(m||i) = C(m)_i$.

Therefore, the output $y = C(m)$ is correct.

μ-Selective-security states that for every ensemble of circuits in $\mathsf{NC}^{1,T,n}$ and pair of messages of length $n(\lambda)$, $\{C_\lambda, m_{0,\lambda}, m_{1,\lambda}\}$ such that $C_\lambda(m_{0,\lambda}) = C_\lambda(m_{1,\lambda})$, the following ensembles of distributions $\{D_{0,\lambda}\}$ and $\{D_{1,\lambda}\}$ are μ-indistinguishable.

$$D_{b,\lambda} = \left(\begin{array}{c} (mpk, msk) \leftarrow \mathsf{FE.Setup}(1^\lambda, T(\lambda)) \\ sk \leftarrow \mathsf{FE.KeyGen}(msk, C_\lambda) \\ c \leftarrow \mathsf{FE.Enc}(1^\lambda, m_{b,\lambda}) \end{array} : mpk, sk, c \right)$$

Below, we show that the computational security of bFE, RE, PPRF, and $i\mathcal{O}^{T,n}$ implies the computational selective security of FE. The proof w.r.t. sub-exponential security is syntactically identical.

Fix a security parameter λ, $C = C_\lambda$, $m_b = m_{b,\lambda}$. By construction of FE, the distribution $D_b = D_{b,\lambda}$ is

$$D_b = \left(\begin{array}{c} (mpk, msk) \leftarrow \mathsf{bFE.Setup}(1^\lambda, T') \\ sk_{\bar{C}} \leftarrow \mathsf{bFE.KeyGen}(msk, \bar{C}) \\ K \xleftarrow{\$} \mathsf{PRF.Gen}(1^\lambda) \\ P_b = P[\lambda, T, n, mpk, 0, K, 0^\lambda, 0^\kappa, m_b] \end{array} : mpk,\ sk_{\bar{C}},\ i\mathcal{O}^{T,n}(1^\lambda, P_b) \right) . \qquad (1)$$

Therefore, it boils down to show that $i\mathcal{O}^{T,n}(1^\lambda, P_0)$ and $i\mathcal{O}^{T,n}(1^\lambda, P_1)$ are indistinguishable, given auxiliary information $mpk, sk_{\bar{C}}$. At a high-level, this follows from the same proof strategy for obfuscating probabilistic circuits abstracted by [CLTV15], which showed that using sub-exp secure IO for $\mathsf{P/poly}$ and puncturable PRF, one can obfuscate, in an indistinguishable way, probabilistic circuits whose outputs are indistinguishable for every input. However, here, we require a more fine-grained proof, using only IO for special-purpose circuits (instead of $\mathsf{P/poly}$), and relying only on its computationally security (instead of sub-exp security). For completeness, we provide a formal proof below.

Towards showing the indistinguishability of $\{D_0\}$ and $\{D_1\}$, we consider a sequence of hybrid distributions $D^b_{i^*}$ for $i^* \in \{0, \cdots T+1\}$ and $b \in \{0,1\}$ described below. We will frequently denote by $\hat{\Pi}_{i,b}$ an RE encoding of the computation that encrypts message $m_b||i$, using bFE and pseudo-random coins, more precisely,

$$\hat{\Pi}_{i,b} = \mathsf{RE.Enc}\left(1^\lambda,\ \mathsf{bFE.Enc},\ (mpk,\ m_b||i\ ;\ R_i)\ ;\ R'_i\right)\ ,\ \text{where } (R_i||R'_i) = \mathsf{F}(K,i) \tag{2}$$

Hybrid Distribution $D^b_{i^*}$ for $i^* \in \{0, \cdots T+1\}$: This distribution is identical to D_0, except that, instead of obfuscating circuit P_0 (as in equation (1)), it obfuscates another special-purpose circuit $P_{i^*,b}$ that differs at the following aspects: (1) $P_{i^*,b}$ is hardwired with a threshold i^* (instead of 0), and a PRF key $K(-i^*)$ punctured at point i^* (instead of $K(-0)$); and (ii) it outputs encoding $\hat{\Pi}_{i,1}$ for every input $i < i^*$, encoding $\hat{\Pi}_{i,0}$ for every $i > i^*$, and a hardwired encoding $\hat{\Pi} = \hat{\Pi}_{i^*,b}$ for $i = i^*$. More precisely, the distribution is

$$D_{i^*,b} = \Big(\ P_{i^*,b} = P[\lambda, T, n, mpk, \underline{i^*, K(-i^*), m_1, \hat{\Pi}_{i^*,b}}, m_0]\ \ :\ mpk,\ sk_{\bar{C}},\ i\mathcal{O}^{T,n}(1^\lambda, P_{i^*,b})\Big) \tag{3}$$

where (mpk, msk), $sk_{\bar{C}}$, and K are sampled identically as in equation (1), and $\hat{\Pi}_{i^*,b}$ is sampled as in equation (2).

Given the above hybrid distributions, to show the indistinguishability of $\{D_0\}$ and $\{D_1\}$, it suffices to prove the following claim, which shows the indistinguishability of every neighboring hybrid distributions.

Claim 2. *The following distribution ensembles are indistinguishable.*

1. *Ensembles $\{D_0\}$ and $\{D_{0,0}\}$ are indistinguishable.*
2. *For every $i^* \in \{0, \cdots, T\}$,ensembles $\{D_{i^*,0}\}$ and $\{D_{i^*,1}\}$ are indistinguishable.*
3. *For every $i^* \in \{0, \cdots, T\}$, ensembles $\{D_{i^*,1}\}$ and $\{D_{i^*+1,0}\}$ are indistinguishable.*
4. *Ensembles $\{D_{T+1,0}\}$ and $\{D_1\}$ are indistinguishable.*

Proof. All distributions listed in the lemma only differ at which circuits are obfuscated. We argue their indistinguishability in sequence:

1. The first indistinguishability follows from the indistinguishability of $i\mathcal{O}^{T,n}$.

   The circuits P_0 and $P_{0,0}$ obfuscated in distributions D_0 and D_{00} have the same functionality: For every input $i \in [T]$, both circuits output the encoding $\hat{\Pi}_{i,0}$ (because, only the flag $b_{i,>}$ would be set to 1; see Figure 1). Thus by the security of $i\mathcal{O}^{T,n}$ for special purpose circuits, $\{D_0\}$ and $\{D_{0,0}\}$ are indistinguishable.

2. The second indistinguishability follows from the security of PPRF, bFE and RE.

   The circuits $P_{i^*,0}$ and $P_{i^*,1}$ obfuscated in distributions $D_{i^*,0}$ and $D_{i^*,1}$ differ only at the hardwired encoding—$\hat{\Pi}_{i^*,0}$ in the former and $\hat{\Pi}_{i^*,1}$ in the latter. Thus, to show the indistinguishability of $D_{i^*,0}$ and $D_{i^*,1}$, it suffices to show the following:

$$\left\{mpk,\ \mathsf{sk}_{\bar{C}},\ \hat{\Pi}_{i^*,0}\right\} \approx \left\{mpk,\ \mathsf{sk}_{\bar{C}},\ \hat{\Pi}_{i^*,1}\right\}$$

   If follows from the pseudo-randomness of PPRF at punctured point i^* that encoding $\hat{\Pi}_{i^*,b}$ is indistinguishable from $\hat{\Pi}'_{i^*,b}$, which encodes the same computation as $\hat{\Pi}_{i^*,b}$ does but using truly random coins. Therefore, it suffices to prove that

$$\begin{aligned}&\left\{mpk,\ \mathsf{sk}_{\bar{C}},\ \hat{\Pi}'_{i^*,0}\right\} \approx \left\{mpk,\ \mathsf{sk}_{\bar{C}},\ \hat{\Pi}'_{i^*,1}\right\}\ ,\\ &\text{where } \hat{\Pi}'_{i^*,b} = \mathsf{RE.Enc}\left(1^\lambda,\ \mathsf{bFE.Enc},\ (mpk,\ m_b||i^*\ ;\ R_{i^*})\ ;\ R'_{i^*}\right)\ ,\\ &\text{and } R_{i^*}||R'_{i^*} \xleftarrow{\$} \{0,1\}^{\mathrm{poly}(\lambda)}\ .\end{aligned}$$

   The output of encoding $\hat{\Pi}'_{i^*,b}$ is $c_{i^*,b} = \mathsf{bFE.Enc}(mpk,\ m_b||i^*\ ;\ R_{i^*})$, a freshly generated ciphertext of message $m_b||i^*$. Since the two challenge messages m_0, m_1 have the same output $C(m_0) = C(m_1)$, we have $\bar{C}(m_0||i^*) = C(m_0)_{i^*} = C(m_1)_{i^*} = \bar{C}(m_1||i^*)$. Therefore, it follows from the selective security of bFE that ciphertext $c_{i^*,b}$ is indistinguishable even given the master public key mpk, and secret key $\mathsf{sk}_{\bar{C}}$ for circuit $\bar{C}$. That is,

$$\begin{aligned}&\{mpk,\ \mathsf{sk}_{\bar{C}},\ c_{i^*,0} = \mathsf{bFE.Enc}(mpk,\ m_0||i^*\ ;\ R_{i^*})\}\\ &\qquad\qquad \approx \left\{mpk,\ \mathsf{sk}_{\bar{C}},\ \hat{\Pi}_{i^*,1},\ c_{i^*,1} = \mathsf{bFE.Enc}(mpk,\ m_1||i^*\ ;\ R_{i^*})\right\}\end{aligned}$$

   Then, it further follows from the security of RE that the encoding $\hat{\Pi}'_{i^*,b}$ is also indistinguishable, given mpk and $\mathsf{sk}_{\bar{C}}$, which concludes the proof.

3. The third indistinguishability follows from the indistinguishability of $i\mathcal{O}^{T,n}$.

   By construction, circuits $P_{i^*,1}$ and $P_{i^*+1,0}$ obfuscated in $D_{i^*,1}$ and $D_{i^*+1,0}$ only differ at how outputs for inputs i^* and i^*+1 are computed. For i^*, the former directly outputs the hardwired encoding $\hat{\Pi}_{i^*,1}$, whereas the latter internally computes the same encoding. For i^*+1, the former internally computes $\hat{\Pi}_{i^*+1,0}$, whereas the latter outputs the hardwired encoding $\hat{\Pi}_{i^*+1,0}$. Nevertheless, since all outputs are identical, by the security of $i\mathcal{O}^{T,n}$ for special purpose circuits, $\{D_{i^*,1}\}$ and $\{D_{i^*+1,0}\}$ are indistinguishable.

4. The fourth indistinguishability follows from the indistinguishability of $i\mathcal{O}^{T,n}$.

   By construction, circuits $P_{T+1,0}$ and P_1 obfuscated in $D_{T+1,0}$ and D_1 have the same functionality: For every input $i \in [T]$, both circuits output the encoding $\hat{\Pi}_{i,1}$ (because, only the flag $b_{i,<}$ would be set to 1; see Figure 1). Thus by the security of $i\mathcal{O}^{T,n}$, $\{D_{T+1,0}\}$ and $\{D_1\}$ are indistinguishable.

□

3.3.2 Setting Parameters

Recall that in the special purpose circuits, the length of the hardwired encoding $\hat{\Pi}$ and the size of the special purpose circuits are padded to some polynomials $\eta'(\lambda, n, \log T)$ and $\eta(\lambda, n, \log T)$ respectively. We now specify them.

Setting η'. For the security proof of FE (especially Claim 2) to go through, we must set η' to be the maximal length of $\hat{\Pi}_{i^*,b}$ for all possible $i^* \in \{0, \cdots, T+1\}$ and $b \in \{0,1\}$ (see equation 2). Since $\hat{\Pi}_{i^*,b}$ is an RE encoding of the computation $\mathsf{bFE.Enc}(mpk, m_b || i^*; R_i)$, its generation time is bounded by $\mathsf{Time}_{\mathsf{RE.Enc}}(\lambda, \mathsf{Time}_{\mathsf{bFE.Enc}}(\lambda, |m_b| + |i^*|))$. Hence, we set

$$\eta'(\lambda, n, \log T) = \mathsf{Time}_{\mathsf{RE.Enc}}(\lambda, \mathsf{Time}_{\mathsf{bFE.Enc}}(\lambda, n + \log T)) \ .$$

Setting η. In order to apply the security of IO in the proof of Claim 2, we pad *every* special purpose circuit $P[\lambda, T, n, mpk, i^*, K, m_<, \hat{\Pi}, m_>]$ to be of the maximal size of all circuits of form $P[\lambda, T, n, *, *, *, m'_<, *, m'_>]$, where $m'_<, m'_>$ have the same length n as $m_<$ and $m_>$ and all wildcard values satisfy the length constraints specified in Figure 1 and above.

It suffices to analyze the worst-case run-time of these special purpose circuits. The first and third steps of P run in time $\mathsf{poly}(\lambda)$, and $\eta'(\lambda, n, \log T)$ respectively. The second and fourth steps generate an RE encoding of an encryption of $\mathsf{bFE.Enc}$, by the same analysis above, their run-time is bound by $\eta'(\lambda, n, \log T)$. Therefore, overall the maximal size is bounded by some polynomial $\eta(\lambda, n, \log T)$.

3.4 Obtaining IO for $\mathsf{P/poly}$

By the construction of FE scheme $\mathsf{FE}^{T,n}$ for $\mathsf{NC}^{1,T,n}$ in Section 3.3, we immediately have the following lemma:

Lemma 3. *Assume the same underlying primitives as Proposition 5. Suppose there is a family of IO schemes $\{i\mathcal{O}^{T,n}\}$ for $\{\{\mathcal{P}_\lambda^{T,n}\}\}$ with universal efficiency, that is,*

$$\mathsf{Time}_{i\mathcal{O}^{T,n}}(1^\lambda, C) \le p(\lambda, |C|) \ , \ \textit{where } p \textit{ is a universal polynomial.}$$

Then, there is a family of FE schemes $\{\mathsf{FE}^{T,n}\}$ for $\{\mathsf{NC}^{1,T,n}\}$ with the following encryption efficiency

$$\mathsf{Time}_{\mathsf{FE.Enc}^{T,n}}(mpk, m) \le p(\lambda, q(\lambda, n(\lambda), \log T(\lambda))) \ , \ \textit{where } q \textit{ is a universal polynomial.}$$

Clearly, this family of FE schemes $\{\mathsf{FE}^{T,n}\}$ gives a compact FE scheme for $\mathsf{NC}^1 = \{\mathsf{NC}^{1,T,n}\}$, and hence already implies IO for $\mathsf{P/poly}$ by Theorem 3 shown in [AJ15, BV15a, ABSV15]. We further examine their results, and observe that for any compact FE scheme, there exist some sufficiently large polynomials T^* and n^*, such that, the resulting IO for $\mathsf{P/poly}$ only uses the FE scheme to generate keys for NC^1 circuits with time bound $T^*(\lambda)$ and input length bound $n^*(\lambda)$. More precisely, we observe the more refined results of [AJ15, BV15a, ABSV15]

Theorem 4 (Refined version of Theorem 3, implicit in [BV15a, AJ15, ABSV15])**.** *Assume the existence of a sub-exponentially secure weak PRF in NC^1, and a (single-query, public-key) FE scheme for $\mathsf{NC}^{1,T,n}$, with encryption time bounded by $T(\lambda)^{1-\varepsilon}$, for sufficiently large polynomials n and T. Then, there exists an indistinguishability obfuscator for $\mathsf{P/poly}$.*

Fix any constant ε. By Lemma 3, for any two sufficiently large polynomials T^*, n^* that satisfy the following condition, the FE scheme FE^{T^*,n^*} constructed from $i\mathcal{O}^{T^*,n^*}$ satisfy the premise of Theorem 4, in particular, the encryption time is smaller than $T^*(\lambda)^{1-\varepsilon}$.

$$p(\lambda, q(\lambda, n^*(\lambda), \log T^*(\lambda))) \leq T^*(\lambda)^{1-\varepsilon}$$

Hence, by Theorem 4, $i\mathcal{O}^{T^*,n^*}$ suffices for building IO for $\mathsf{P/poly}$. This concludes Proposition 5.

For completeness we provide a proof sketch of Theorem 4 in Appendix 8.

4 Special-Purpose Circuits in Constant Degree

Assuming a constant-degree PRG, we show how to implement the special-purpose circuits in Figure 1 using constant-degree arithmetic circuits.

Proposition 6. *Instantiated with a constant-degree PRG, the class of special-purpose circuits $\{\mathcal{P}_\lambda^{T,n}\}$ in Figure 1 has universal arithmetic circuits $\{U_\lambda\}$ of constant degree* $\deg$ *and size $u(\lambda, n, \log T)$ for a universal polynomial u independent of T, n. Thus, the family of special-purpose circuit classes $\{\{\mathcal{P}_\lambda^{T,n}\}\}$ has constant-degree.*

Jumping ahead, when later trying to construct an special-purpose IO scheme for the class of circuit in the above proposition, we not only need to rely on the fact that the circuits have constant degree, but also that they have additional structure, namely, having constant type degree (introduced later). Here we first prove the weaker constant degree property, since it is an important step towards proving the stronger constant type degree property later. Moreover, this property is interesting on its own, and it already leads to the following strong bootstrapping theorems for IO. By Proposition 5, and the fact that all underlying primitives of the Proposition are implied by the hardness of LWE (see the discussion in Section 3.1), we have,

Theorem 5 (Bootstrapping IO for constant degree circuits)**.** *Assume sub-exponential hardness of LWE, and the existence of a sub-exponentially secure constant-degree PRG. There exist a family of circuit classes of constant degree, such that, IO for that family with universal efficiency can be bootstrapped into IO for* $\mathsf{P/poly}$*.*

As discussed in the introduction, the above proposition and theorem can also be generalized to use arbitrary PRG not necessary computable in constant degree. See Remark 1 for more details.

4.1 Overview

The class $\{\mathcal{P}_\lambda^{T,n}\}$ consists of special purpose circuits of the form $P[\lambda, T, n, \star_1](\star_2)$, where $T = T(\lambda)$ and $n = n(\lambda)$, where $\star_1$ represents the rest of the constants (including $mpk, i^*, K, m_<, \hat{\Pi}, m_>$) and $\star_2$ represents the input i. By viewing the rest of the constants as a description of the circuit, $U(\star_2, \star_1) = P[\lambda, T, n, \star_1](\star_2)$ can be viewed as the universal circuit of family $\mathcal{P}_\lambda^{T,n}$. Hence, towards the proposition, we only need to argue that $P[\lambda, T, n, \star](\star)$ can be implemented by an arithmetic circuit of constant degree and size $\mathrm{poly}(\lambda, n, \log T)$.

The computation of P can be broken down into three parts: i) Evaluating the PPRF in Step 1, ii) performing comparison between i and i^*, and iii) depending on the outcome of comparison, potentially compute a RE encoding in NC^0. By definition of RE in NC^0, part iii) has constant degree. The challenges lie in implementing Part i) and ii) in constant degree. More specifically,

Challenge 1: Let $b_{i,<}$, $b_{i,=}$, $b_{i,>}$ be decision bits indicating whether the input i is smaller than, equal to, or greater than the hardwired threshold i^*. Since $i \in [T]$ and $i^* \in \{0, \cdots, T+1\}$, their binary representation has logarithmic length $l = \lceil \log(T+2) \rceil$. Under binary representation, the straightforward way of computing these decision bits also requires logarithmic $O(l) = O(\log T)$ multiplications. E.g., equality testing can be done as $b_{i,=} = \prod_{j \in [l]}(1 - (i_j - i^*_j)^2)$ (over any ring, where i_j and i^*_j are the j^{th} bit of i and i^*).

Challenge 2: The state-of-the-art PPRF scheme [BV15b] has an evaluation algorithm in NC^1 (assuming LWE), far from computable in constant degree. Even without the puncturing functionality, standard PRFs cannot be computed in constant degree, or even AC^0, since such functions are learnable [LMN89].

Towards overcoming above challenges, we rely on the simple, but powerful, observation is that in our special-purpose circuits, the input i and threshold i^* both belong to a polynomial-sized set $\{0, \cdots, T+1\}$ (T by definition is polynomial in λ). This allows us to switch the representation of i and i^* from binary strings of length $O(\log T)$ to *strings of constant length over a polynomial-sized alphabet*, presented below.

New Input Representation Instead of using binary alphabet, we represent the input $i \in [T]$, as well as the hardwired threshold $i^* \in \{0, \cdots, T+1\}$, using an alphabet Σ consisting of a polynomial number of vectors of length λ,

$$\Sigma = \{\mathbf{e}_0, \cdots, \mathbf{e}_\lambda\} \ , \tag{4}$$

where $\mathbf{e}_j$ for $j \in \{0, \cdots, \lambda\}$ contains 1 at position j and 0 everywhere else (in particular, $\mathbf{e}_0$ is the all 0 vector). Since T is polynomial in λ, there is a minimal constant, c such that, i (as well as i^*) can be divided into c blocks of length $\lfloor \log(\lambda+1) \rfloor$, denoted as $i = \mathsf{i}_1 || \mathsf{i}_2 || \cdots || \mathsf{i}_c$. Therefore, using alphabet Σ,

$$i \overset{\Sigma}{=} \mathbf{e}_{\mathsf{i}_1} || \cdots || \mathbf{e}_{\mathsf{i}_c} \ , \ \text{with length } |i|_\Sigma = c\lambda \ ,$$

where $a \overset{\Sigma}{=} b$ denote that b is the representation of a using alphabet Σ, and $|a|_\Sigma$ denote the number of bits needed in order to describe the representation over Σ.

We sketch how to resolve the two challenges, using the new representation.

Overcoming the first challenge: consider the simple task of testing equality of one block, i_k and i^*_k—flag $b^k_{i,=}$ is set to 1 iff $\mathsf{i}_k = \mathsf{i}^*_k$. With the new representation, this equality can be tested by simply computing $b^k_{i,=} = \mathbf{e}_{\mathsf{i}_k} \cdot \mathbf{e}_{\mathsf{i}^*_k}$ in degree two. Moreover, after testing equality of all blocks, which can be done in parallel, the equality between i and i^* can be computed as $b_{i,=} = \prod_{k \in [c]} b^k_{i,=}$ in constant degree c. Testing other relations, smaller than and greater than, between i and i^* can be performed similarly. See Section 4.2 for details.

Overcoming the second challenge: To circumvent the impossibility results, we leverage the fact that we only need to construct a PPRF for a special polynomial-sized domain σ^c. Assume the existence of a constant-degree PRG with super-linear stretch. The most natural idea is to construct a PPRF using the GGM PRF tree [GGM86] as done in previous constructions of PPRF [BW13, BGI14, KPTZ13]. Clearly, the degree of the PPRF evaluation grows exponentially with the depth of the tree. Therefore, we can tolerate at most a constant depth. Fortunately, our domain is of polynomial size, and if we use a high-degree GGM tree, where each node has λ children, the depth is constant $O(c)$. However, an issue arises when using high-degree tree. Recall that the evaluation of the GGM PRF requires following the path leading to the leaf indexed by the input; at a particular node, the evaluator needs to choose the appropriate child in the next layer. When the tree has degree λ, choosing a child corresponds to the indexing function called the multiplexer $\mathsf{mux}(\vec{v}, j) = \vec{v}_j$, which has at least depth $\Omega(\log|\vec{v}|)$ when j is represented in binary. But, again thanks to our new input presentation $j \overset{\Sigma}{=} \mathbf{e}_j$, mux can be implemented as $\vec{v} \cdot \mathbf{e}_j$ in degree 2. See Section 4.3 for details on the PPRF.

Finally, we put all pieces together in Section 4.4. Our final implementation of special purpose circuits had degree of order $\exp(\log_\lambda(Tn))$.

4.2 Performing Comparisons in Constant-Degree

We show how to perform various comparison between i and i^* represented using the new input representation in constant degree. Towards this, we first show how to perform comparison over any single block of i and i^* in degree 2. For any $k \in [c]$, let $b^k_{i,<}$, $b^k_{i,=}$, $b^k_{i,>}$ be flags indicating whether the k^{th} block of i, i_k, is smaller than, or equal to, or greater than the corresponding block of i^*, i^*_k; they can be computed as follows:

- $b^k_{i,=}$ can be computed as the inner product $b^k_{i,=} = \mathbf{e}_{\mathsf{i}_k} \cdot \mathbf{e}_{\mathsf{i}^*_k}$.
- $b^k_{i,<}$ can be computed as the inner product $b^k_{i,<} = \mathbf{e}_{\mathsf{i}_k} \cdot \mathbf{e}_{<\mathsf{i}^*_k}$, where $\mathbf{e}_{<\mathsf{i}^*_k}$ denote the vector that contains 1s in the first $\mathsf{i}^*_k - 1$ positions, and 0s in the rest.
- $b^k_{i,>}$ can be similarly computed as the inner product $b^k_{i,>} = \mathbf{e}_{\mathsf{i}_k} \cdot \mathbf{e}_{>\mathsf{i}^*_k}$, where $\mathbf{e}_{>\mathsf{i}^*_k}$ denote the vector that contains 0s in the first i^*_k positions, and 1s in the rest.

Next, performing comparison over entire i and i^* involves congregating the results of comparisons over individual blocks, which can be done using only a constant number $O(c)$ of multiplications as described in Figure 2.

4.3 PRF Evaluation in Constant-Degree

The special purpose circuits require a PPRF function with input domain $\{0, \cdots, T\}$, key domain $\{0,1\}^\lambda$, and range $\{0,1\}^{L(\lambda)}$ for $L(\lambda)$ long enough to supply the random coins for bFE and RE; hence $L(\lambda) = \text{poly}(\lambda, n, \log T)$. The following lemma provides such a PPRF in constant degree.

Lemma 4. *Assume the existence of a degree-d PRG with $\lambda^{1+\varepsilon}$-stretch for some constant $d \in \mathbb{N}$ and $\varepsilon > 0$. For every polynomial D and L, there is a degree $\deg'$ PPRF scheme with input domain $\{0, \cdots, D(\lambda)\}$, key domain $\{0,1\}^\lambda$, and range $\{0,1\}^{L(\lambda)}$, where $\deg' \in \mathbb{N}$ is some constant depending on d, ε, D and L. Furthermore, if the underlying PRG is subexponentially secure, then so is the PPRF.*

Proof. Let PRG be the PRG in the premise. We first make the observation that it implies a constant-degree PRG scheme qPRG with quadratic stretch: If the stretch of PRG is already more than quadratic, (i.e., $1+\varepsilon \geq 2$) simply truncate the output to length λ^2. Otherwise, iteratively evaluate PRG for a sufficient number $I = \lceil 1/\log(1+\varepsilon) \rceil$ of times to expand the output to length λ^2, that is, $\mathsf{qPRG}(s) = \mathsf{PRG}^I(s)$. The degree of qPRG increases to d^I, still a constant, and the security of qPRG follows from standard argument. Below, we will view the output of qPRG as a vector $\mathbf{v} = \mathbf{v}[1], \cdots, \mathbf{v}[\lambda]$ of λ elements, each $\mathbf{v}[i]$ is a λ-bit binary string.

Furthermore, we observe that to get a PPRF with range $\{0,1\}^{L(\lambda)}$, it suffices to construct one with range $\{0,1\}^\lambda$, since one can always apply PRG iteratively to expand the output to $L(\lambda)$ as argued above.

Performing Comparisons $\mathsf{Compare}(i)$

Constants: a threshold $i^* \in \{0, \cdots, T+1\}$ represented as $i^* \overset{\Sigma}{=} (\mathbf{e}_{\mathsf{i}^*_k})_{k\in[c]}$ together with vectors $(\mathbf{e}_{<\mathsf{i}^*_k}, \mathbf{e}_{>\mathsf{i}^*_k})_{k\in[c]}$.

Input: an input $i \in [T]$ represented as $i \overset{\Sigma}{=} (\mathbf{e}_{\mathsf{i}^*_k})_{k\in[c]}$.

Procedure:

1. For every $k \in [c]$, compute $b^k_{i,=} = \mathbf{e}_{\mathsf{i}_k} \cdot \mathbf{e}_{\mathsf{i}^*_k}$, $b^k_{i,<} = \mathbf{e}_{\mathsf{i}_k} \cdot \mathbf{e}_{<\mathsf{i}^*_k}$, and $b^k_{i,>} = \mathbf{e}_{\mathsf{i}_k} \cdot \mathbf{e}_{>\mathsf{i}^*_k}$.
2. Do the following in parallel:

<u>*Testing* $i = i^*$</u> requires checking whether all blocks are equal. Therefore,

$$b_{i,=} = \prod_{k\in[c]} b^k_{i,=} \ . \tag{5}$$

<u>*Testing* $i < i^*$</u> requires checking whether one of the following cases occur: For some $k \in [c]$, the first $k-1$ blocks of i and i^* are equal, and the k^{th} block of i is smaller than that of i^*. Therefore,

$$b_{i,<} = 1 - \prod_{k\in[c]} \left(1 - \left(\prod_{j<k\in[c]} b^j_{i,=}\right) \times b^k_{i,<}\right) \ . \tag{6}$$

<u>*Testing* $i > i^*$</u> requires checking whether one of the following cases occur: For some $k \in [c]$, the first $k-1$ blocks of i and i^* are equal, and the k^{th} block of i is larger than that of i^*. Therefore,

$$b_{i,>} = 1 - \prod_{k\in[c]} \left(1 - \left(\prod_{j<k\in[c]} b^j_{i,=}\right) \times b^k_{i,>}\right) \ . \tag{7}$$

Figure 2: Performing comparisons between i and i^* in constant degree. Vector $\mathbf{e}_j \in \{0,1\}^\lambda$ has a single 1 at position j and 0 elsewhere, and vector $\mathbf{e}_{<j} \in \{0,1\}^\lambda$ has 1 at positions $< j$, and 0 elsewhere.

Using qPRG, we now construct a PPRF scheme $\mathsf{PPRF} = (\mathsf{PRF.Gen}, \mathsf{PRF.Punc}, \mathsf{F})$ with λ-bit outputs. Since D is a polynomial, there is a minimal integer c such that for all $\lambda \in \mathbb{N}$, $D(\lambda) < \lambda^c$. Fix any security parameter λ, and $D = D(\lambda)$. Our scheme PPRF with input domain $\{0, \cdots, D\}$ represents inputs under alphabet Σ (in equation (4)), or alternatively, the input domain is Σ^c.

Key Generation $\mathsf{PRF.Gen}(1^\lambda)$ samples a random λ-bit string $K \xleftarrow{\$} \{0,1\}^\lambda$.

Key Puncturing $\mathsf{PRF.Punc}(K, i^*)$ sets $K_0 = K$ and computes the following for every $k \in [c]$:

- $\mathbf{v}_k = \mathsf{qPRG}(K_{k-1})$.
- Let $\mathbf{v}_k[\neq i^*_k]$ be the vector identical to $\mathbf{v}_k$, but with the $i^{*\,\text{th}}_k$ element replaced with 0.

Set the punctured key as

$$K(-i^*) = \left(\mathbf{e}_{\mathsf{i}^*_k},\ \mathbf{v}_k[\neq i^*_k]\right)_{k\in[c]}$$

Note that the size of $K(-i^*)$ is bounded by $O(\lambda^2)$.

PRF Evaluation $\mathsf{F}(K(-i^*), i)$ is presented in Figure 3. It is easy to verify that the algorithm indeed has constant-degree.

PRF Evaluation $\mathsf{F}(K(-i^*), i)$

Input: A punctured key $K(-i^*) = (\mathbf{e}_{\mathsf{i}^*_k}, \mathbf{v}_k[\neq i^*_k])_{k\in[c]}$, and an input $i \in \{0, \cdots, D\}$ represented as $i \overset{\Sigma}{=} (\mathbf{e}_{\mathsf{i}_k})_{k\in[c]}$. By definition $i^* \neq i$.

Procedure:

1. For every $k \in [c]$, compute $b^k_{i,=} = \mathbf{e}_{\mathsf{i}_k} \cdot \mathbf{e}_{\mathsf{i}^*_k}$, which indicates whether the k^{th} blocks i^*_k and i_k are equal.
2. For every $k \in [c]$, compute d^k_i indicating whether the following occurs: The first $k-1$ blocks of i and i^* are equal, but the k^{th} block differs.
$$d^k_i = \Big(\prod_{j<k\in[c]} b^j_{i,=}\Big) \times \big(1 - b^k_{i,=}\big) \ .$$
3. For every $k \in [c]$, do:
   - Select the i^{th}_k element in $\mathbf{v}_k[\neq i^*_k]$,
   $$K^k_k = \mathbf{v}_k[\neq i^*_k] \cdot \mathbf{e}_{\mathsf{i}_k} \ .$$
   - For $j = k+1$ to c, compute
   $$\mathbf{w}_j = \mathsf{qPRG}(K^k_{j-1}) \ , \ K^k_j = \mathbf{w}_j \cdot \mathbf{e}_{\mathsf{i}_j} \ .$$
4. Compute the final output
$$y = \Sigma_{k\in[c]}(K^k_c \times d^k_i)$$

In the last two steps, multiplication between a string z and bit b yields $0^{|z|}$ if $b = 0$ and z if $b = 1$, and addition between two strings is bit-wise addition. Inner product between a vector of strings and a vector of bits are defined accordingly.

Figure 3: Constant-degree PRF evaluation

Efficiency and security: The only difference between the above scheme and the original constructions of PPRF based on GGM tree [BW13, BGI14, KPTZ13] is (i) the tree has degree λ instead of degree 2, and (ii) the inputs i and i^* are represented under Σ. For efficiency, the second difference has no impact, since under Σ, the representation of i and i^* are still of fixed polynomial size; the only effect the first difference has is that the punctured key consists of a λ-sized vector per layer of the tree, as opposed to 1 element per layer, but the size of the punctured key is still bounded by a fixed polynomial. For security, the same proof of [BW13, BGI14, KPTZ13] goes through even when the tree has higher degree; we omit details here.

□

4.4 Putting Pieces Together

Given the sub-routine $\mathsf{Compare}$ and a constant-degree PPRF scheme PPRF with domain $\{0, \cdots, T+1\}$ and appropriate output length $L(\lambda) = \text{poly}(\lambda, n, \log T)$, a constant-degree implementation the special-purpose circuits is presented in Figure 4, where Step 1 and 2 evaluate the new functions

Constant Degree Circuit $P[\lambda, T, n, mpk, i^*, K, m_<, \hat{\Pi}, m_>]$

Constants: $\lambda, T, mpk, m_<, \hat{\Pi}, m_>$ are defined as in Figure 1; $i^* \in \{0, \cdots, T+1\}$ is represented as $i^* \overset{\Sigma}{=} (\mathbf{e}_{\mathsf{i}^*_k})_{k\in[c]}$, together with vectors $(\mathbf{e}_{<\mathsf{i}^*_k}, \mathbf{e}_{>\mathsf{i}^*_k})_{k\in[c]}$; K is a punctured key of a constant degree PPRF PPRF.

Input: index $i \in [T]$ represented under Σ, that is, $(\mathbf{e}_{\mathsf{i}_k})_{k\in[c]}$.

Procedure:

1. $(R_i || R'_i) = \mathsf{F}(K, (\mathbf{e}_{\mathsf{i}_k})_{k\in[c]})$. (See Figure 3.)
2. $b_{i,<}, b_{i,=}, b_{i,>} = \mathsf{Compare}[i^*]((\mathbf{e}_{\mathsf{i}_k})_{k\in[c]})$. (See Figure 2.)
3. For $\star \in \{<, >\}$, compute
   $\hat{\Pi}_{i,\star} = \mathsf{RE.Enc}\left(1^\lambda,\ \mathsf{bFE.Enc},\ (mpk,\ m_\star || (\mathbf{e}_{\mathsf{i}_k})_{k\in[c]});\ R_i);\ R'_i\right)$.
4. Output $\hat{\Pi}_i = \hat{\Pi}_{i,<} \times b_{i,<} + \hat{\Pi} \times b_{i,=} + \hat{\Pi}_{i,>} \times b_{i,>}$.

Padding: The hardwired encoding $\hat{\Pi}$ is padded to be of length $\bar{\eta}'(\lambda, n, \log T)$, and the circuit is padded to be of size $\bar{\eta}(\lambda, n, \log T)$, for some polynomials $\bar{\eta}'$ and $\bar{\eta}$ set similarly as in Section 3.3.2.

Figure 4: Special-Purpose Circuit P in Constant Degree

$\mathsf{Compare}$ and PPRF respectively. The choice of which randomized encoding to output, depending on the outcome of comparisons, is made in Step 4 using simple addition and multiplication. Moreover, since the index i is now represented under Σ, each of its appearance in the special purpose circuit (e.g. in Step 3), as well as in the bootstrapping transformation of Proposition 5 is replaced with $(\mathbf{e}_{\mathsf{i}_1}, \cdots, \mathbf{e}_{\mathsf{i}_c})$. Since this representation also has a fixed polynomial size (bounded by λ^2 for sufficiently large λ), all constructions and proofs remain intact.

It is easy to see that the implementation is correct, and furthermore the circuit size of this implementation is still $u(\lambda, n, \log T)$ for some universal polynomial u independent of T, n: In Step 1, the evaluation of the PPRF takes fixed (universal) polynomial time $\mathrm{poly}(\lambda)$, and so is the evaluation of function $\mathsf{Compare}$ in Step 2. The run-time of Step 3 and 4 is determined by that of RE and bFE as before, which again is bounded by a fixed (universal) polynomial $\mathrm{poly}(\lambda, n, \log T)$. Therefore, the worst-case run-time and hence circuit size is bounded by $u(\lambda, n, \log T)$, for some universal polynomial u.

Remark 1 (Generalization to Any PRG)**.** *We generalize Proposition 6 to the case with a general* PRG*. Since* PRG *is only used for constructing the PPRF in Figure 3, and all other steps can be implemented in constant degree, the special purpose circuits have the form* P^{PRG} *of constant degree circuits with black-box access to* PRG*. This gives the more general bootstrapping theorem in Proposition 4 discussed in the introduction. In addition, it is easy to see that since the PPRF is constructed as a tree, where each node corresponds to a PRG evaluation, if PRG has degree* $d(\lambda)$*, then the PPRF has degree* $\mathrm{poly}(d(\lambda)) = d(\lambda)^c$*, where* c *is the depth of the tree, and is a constant.*

5 Graded Encoding Schemes and Ideal Graded Encoding Oracle

We recall the definitions of Graded Encoding (GE) Schemes, ideal Graded Encoding (GE) Oracle, and IO in oracle model. In this work, we consider GE scheme and GE oracle w.r.t. composite order rings from a distribution, and treats the *(multi-)*sets that elements are encoded under as vectors, as in the work of Applebaum and Brakerski [AB15].

Composite-Order Rings: we work with composite-order rings $\mathbb{Z}_P$, where $P = \prod_{i=1}^{\sigma} p_i$ and $p_1, \cdots, p_\sigma$ are distinct co-prime numbers. By the Chinese Remainder Theorem (CRT), $\mathbb{Z}_P \cong \mathbb{Z}_{p_1} \times \cdots \times \mathbb{Z}_{p_\sigma}$. The CRT representation of elements a in $\mathbb{Z}_P$ is $a \cong a_1, \cdots a_\sigma$; we also denote by $a[\![i]\!]$ the component $a_i = a \mod p_i$.

Sets and Levels: Elements in $\mathbb{Z}_P$ are encoded under multi-sets over a universe $[\tau]$, which are represented as a vector in $\mathbb{N}^\tau$. There is a natural partial ordering on vectors in $\mathbb{N}^\tau$: $\mathbf{v} \leq \mathbf{w}$ if for all $i \in [\tau]$, it holds that $\mathbf{v}[i] \leq \mathbf{w}[i]$. If there is a coordinate $i \in [\tau]$ in which $\mathbf{v}[i] > \mathbf{w}[i]$, then we say $\mathbf{v}[i] \not\leq \mathbf{w}[i]$. We also call these sets *levels*.

5.1 Graded Encoding Schemes

Our GE scheme for composite order groups follows the definition of [AB15], which in turn is based on the definition of GE scheme in [GGH13a] and follow-up works. The exact formalization below differs slightly from that in [AB15]: Following [GGH+13b, BGK+14] and the notion of multi-linear jigsaw puzzles from [GGH+13b], our definition allows anyone with the secret parameters to encode any elements at any level, whereas in [AB15], only random ring or sub-ring elements can be encoded. Our construction of IO can also work with the more restricted interface of [AB15]. For simplicity, we use the simpler interface.

Definition 12 (Graded Encoding Scheme). *Let $\mathcal{R}$ be a ring, and let $\mathbf{v}_{zt} \in \mathbb{N}^\tau$ be an integer vector of dimension $\tau \in \mathbb{N}$. A graded encoding scheme for $\mathcal{R}$, $\mathbf{v}_{zt}$ is a collection of sets $\{[\alpha]_{\mathbf{v}} \subseteq \{0,1\}^* \ : \ \mathbf{v} \in \mathbb{N}^\tau, \mathbf{v} \leq \mathbf{v}_{zt}, \alpha \in \mathcal{R}\}$, with the following properties:*

1. *For every index $\mathbf{v} \leq \mathbf{v}_{zt}$, the sets $[\alpha]_{\mathbf{v}} : \alpha \in \mathcal{R}$ are disjoint. We slightly abuse notation and often denote $a = [\alpha]_{\mathbf{v}}$ instead of $a \in [\alpha]_{\mathbf{v}}$.*

2. *There are associative binary operations $+$ and $-$ such that for all $\mathbf{v} \in \{0,1\}^\tau$, $\alpha_1, \alpha_2 \in \mathcal{R}$ and all $u_1 = [\alpha_1]_{\mathbf{v}}$ and $u_2 = [\alpha_2]_{\mathbf{v}}$: $u_1 + u_2 = [\alpha_1 + \alpha_2]_{\mathbf{v}}$ and $u_1 - u_2 = [\alpha_1 - \alpha_2]_{\mathbf{v}}$, where $\alpha_1 + \alpha_2$ and $\alpha_1 - \alpha_2$ are addition and subtraction in $\mathcal{R}$.*

3. *There is an associative binary operation $\times$ such that for all $\mathbf{v}_1, \mathbf{v}_2 \in \mathbb{N}^\tau$ such that $\mathbf{v}_1 + \mathbf{v}_2 \leq \mathbf{v}_{zt}$, for all $\alpha_1, \alpha_2 \in \mathcal{R}$ and for all $u_1 = [\alpha_1]_{\mathbf{v}_1}$, $u_2 = [\alpha_2]_{\mathbf{v}_2}$, it holds that $u_1 \times u_2 = [\alpha_1 \cdot \alpha_2]_{\mathbf{v}_1 + \mathbf{v}_2}$, where $\alpha_1 \cdot \alpha_2$ is multiplication in $\mathcal{R}$.*

As in [AB15], we consider GE scheme for composite order rings $\mathcal{R}$, which is potentially sampled from a distribution (as opposed to be fixed a priori). Below, define efficient procedures associated with such a scheme.

Definition 13 (Efficient Procedures for Graded Encoding Schemes). *A graded encoding schemes* GES *is associated with a tuple of PPT algorithms* $(\mathsf{InstGen}, \mathsf{Encode}, \mathsf{Add}, \mathsf{Sub}, \mathsf{Mult}, \mathsf{isZero})$ *which behaves as follows:*

- *Instance Generation:* $(\mathsf{pp}, \mathsf{sp}) \xleftarrow{\$} \mathsf{InstGen}(1^\lambda, 1^\sigma, 1^k, 1^{\mathbf{v}_{zt}})$ *on input a security parameter λ, the number σ of subrings, and multilinearity parameter k, and a zero-test set $\mathbf{v}_{zt}$, outputs a secret parameter* sp *that describes a $(\mathcal{R}, \mathbf{v}_{zt})$-graded encoding scheme, where $\mathcal{R} = \mathbb{Z}_P \cong \mathbb{Z}_{p_1}, \cdots \mathbb{Z}_{p_\sigma}$ with pairwise co-prime numbers p_i, and a public parameter* pp *that is a subset of* sp *only sufficient for performing addition, subtraction, and multiplication.*

  Hidden Ring: We remark that $\mathcal{R}$ is potentially sampled according to a distribution and kept hidden, in particular, it is not included in pp.

- *Encoding:* Encode *on input* sp*, a ring element* $\alpha \in \mathcal{R}$*, and level* $\mathbf{v} \in \mathbb{N}^\tau$ *outputs an encoding in* $[\alpha]_{\mathbf{v}}$.

- *Addition:* Add *on input* pp*, encodings* $A_1 = [\alpha_1]_{\mathbf{v}_1}$ *and* $A_2 = [\alpha_2]_{\mathbf{v}_2}$*, outputs* $B = [\alpha_1 + \alpha_2]_{\mathbf{v}}$ *if* $\mathbf{v}_1 = \mathbf{v}_2 = \mathbf{v}$ *and* $\perp$ *otherwise.*

- *Negation:* Sub *on input* pp *and an encoding* $A = [\alpha]_{\mathbf{v}}$*, outputs* $B = [-\alpha]_{\mathbf{v}}$.

- *Multiplication:* Mult *on input* pp*, encodings* $A_1 = [\alpha_1]_{\mathbf{v}_1}$ *and* $A_2 = [\alpha_2]_{\mathbf{v}_2}$*, outputs* $B = [\alpha_1 \cdot \alpha_2]_{\mathbf{v}_1+\mathbf{v}_2}$ *if* $\mathbf{v}_1 + \mathbf{v}_2 \leq \mathbf{v}_{zt}$ *and* $\perp$ *otherwise.*

- *Zero Testing:* isZero *on input* pp*, and encoding* $A = [\alpha]_{\mathbf{v}}$ *outputs 1 if and only if* $\alpha = 0$ *and* $\mathbf{v} = \mathbf{v}_{zt}$.

Noisy Encodings. In known candidate constructions, encodings are noisy and the noise level increases with addition and multiplication operations, so one has to be careful not to go over a specified noise bound. However, the parameters are set so as to support poly(λ) additions and $O(k)$ multiplications, where k is the multilinearity parameter received by InstGen. This will be sufficient for our purposes and we therefore ignore noise management throughout.

We remark that in previous works, the multilinearity parameter is set to $|\mathbf{v}_{zt}|_1$, because any computation with degree higher than $|\mathbf{v}_{zt}|_1$ produces encodings with level exceeding $\mathbf{v}_{zt}$, and hence are not "supported". In this work, we separate the multilinearity parameter from $|\mathbf{v}_{zt}|_1$, since the degree "supported" is much smaller than it.

5.2 The Level Function and Level-Respecting Arithmetic Circuits

Given that GE scheme associates computation with levels and imposes constraints on how levels can be manipulated. It would be instrumental to define the following level function level and level-respecting arithmetic circuits. The latter is termed set-respecting circuits in previous work [PST14a]; in this work, we represent sets as vectors and call them levels, and hence the name level-respecting circuits for consistency.

Definition 14 (The level function level)**.** *For every sequence* $\vec{\mathbf{v}}$ *of levels and an arithmetic circuit* C*, we define an assignment of levels to every wire* w *in* C *through the following recursively defined function* $\mathsf{level}(C, \vec{\mathbf{v}}, w)$*:*

- *If* w *is the* i^{th} *input wire, label it with level* $\mathbf{v}[i]$.

- *If* w *is the output wire of an addition (+) or subtraction (-) gate in* C *with input wires* u_1 *and* u_2*, and* $\mathsf{level}(C, \vec{\mathbf{v}}, u_1) = \mathsf{level}(C, \vec{\mathbf{v}}, u_2) = \mathbf{v} \neq \perp$*, label* w *with level* $\mathbf{v}$.

- *If* w *is the output wire of a multiplication gate in* C *with input wires* u_1 *and* u_2*, and* $\mathsf{level}(C, \vec{\mathbf{v}}, u_1) = \mathbf{v}_1 \neq \perp$ *and* $\mathsf{level}(C, \vec{\mathbf{v}}, u_2) = \mathbf{v}_2 \neq \perp$*, label* w *with level* $\mathbf{v}_1 + \mathbf{v}_2$.

- *In all other cases, label* w *with* $\perp$.

Definition 15 (Level-Respecting Arithmetic Circuits)**.** *We say that an arithmetic circuit* C *is* $\vec{\mathbf{v}}$*-respecting if for every output wire* γ *of* C*,* $\mathsf{level}(C, \vec{\mathbf{v}}, \gamma) \neq \perp$*. Moreover, let* $\mathbf{v}_{zt}$ *be another level, we say that* C *is* $(\vec{\mathbf{v}}, \mathbf{v}_{zt})$*-respecting if for every output wire* γ *of* C*,* $\mathsf{level}(C, \vec{\mathbf{v}}, \gamma) = \mathbf{v}_{zt}$.

5.3 Ideal Graded Encoding Oracle

An ideal graded encoding oracle models the ideal interface of GE scheme; proving security of a construction that uses an ideal graded encoding oracle shows that it is resilient to "generic" attacks that treat encodings as if they were "physical envelopes" on which only legitimate operations can be performed.

Following [AB15], to model GE scheme over rings from a distribution, we consider an ideal graded encoding oracle $\mathcal{O}$ parameterized with a distribution $\tilde{\mathcal{R}}$ over rings. The oracle samples a ring $\mathcal{R} \xleftarrow{\$} \tilde{\mathcal{R}}(1^\lambda)$ and enables players to 1) encode an element $\alpha \in \mathcal{R}$ under a level $\mathbf{v}$, and receive a random "handle" h in return, and to 2) make "legal" zero-test queries on these encodings: a zero-test query is a formal polynomial p on variables $\vec{h}$, which evaluates to `true` iff $p(\vec{v}) = 0$, where for every i, v_i is the value encoded under handle h_i.

In this work, we will consider ideal graded encoding oracles that are *constant-degree*, that is, all legal polynomials that could be zero-tested have constant degree. This notion is recently formulated in [Pas15, MMN15]. We here follow the formalization of [Pas15], where the legality of a query is determined by a *legality-predicate* g: $g(1^\lambda, p, \vec{l})$ outputs 1 if the query is deemed legal, where $\vec{l}$ are the labels corresponding to the handles $\vec{h}$.

Below, we provide the formal definition of ideal graded encoding oracle adapted from [Pas15].

Definition 16 (Ideal Graded Encoding Oracle)**.** *An ideal graded encoding oracle is an ensemble of stateful oracles $\{\mathcal{O}_\lambda^{g,\tilde{\mathcal{R}}}\}$ parameterized by a distribution $\tilde{\mathcal{R}}$ over rings and a legality predicate g, and indexed by a security parameter λ, each responds to queries in the following manner:*

1. *Upon initialization, a ring is sampled $\mathcal{R} \xleftarrow{\$} \tilde{\mathcal{R}}(1^\lambda)$, and sent to the activator.*

2. *Then (and only then), the activator may* adaptively *make any number of queries of the form* $\mathsf{OEnc}(\alpha, \mathbf{v})$*, with $\alpha \in \mathcal{R}$ and $\mathbf{v} \in \mathbb{N}^\tau$; for each such query, $\mathcal{O}_\lambda^{g,\tilde{\mathcal{R}}}$ picks a uniformly random "handle" $h \in \{0,1\}^{\lambda\ell}$ where $\ell = \ell(\lambda)$ is an upper bounded on the length of elements in rings sampled from $\tilde{\mathcal{R}}$, stores the tuple $(\alpha, \mathbf{v}, h)$ in a list $\mathcal{L}_\mathcal{O}$ and returns h.*[2] *(This initialization phase ends if any algorithm other than the activating algorithm makes any query to $\mathcal{O}_\lambda^{g,\tilde{\mathcal{R}}}$, or if the activator makes a non* OEnc *query. Any subsequent* $\mathsf{OEnc}(\cdot,\cdot)$ *queries will be answered with $\perp$.)*

3. *On input a query* $\mathsf{OEval}(p)$ *where p is a formal polynomial over variables $h_1, \ldots, h_m$, each of which is represented as a string of length $\lambda\ell$ (corresponding to some handle), $\mathcal{O}_\lambda^{g,\tilde{\mathcal{R}}}$ does the following:*

   (a) *For each $i \in [m]$, retrieve a tuple $(\alpha_i, \mathbf{v}_i, h_i)$ from the state $\mathcal{L}_\mathcal{O}$; if no such tuple exists, it returns* `false`*.*

   (b) *(Illegal query) If all tuples are retrieved, return* `false` *if $g(1^\lambda, p, \vec{\mathbf{v}}) \neq 1$*

   (c) *(Zero test) Finally, return* `true` *iff $p(\alpha_1, \ldots, \alpha_n) = 0 \mod q$, and* `false` *otherwise.*

4. *(All other queries are answered with $\perp$).*

[2]In particular, even if the same value v is encoded twice (under the same label), we get independently random handles for the two encodings. This model thus considers *randomized* graded encodings. We mention, however, that our results also work for deterministic graded encodings, where the oracle keeps state also during the encoding phase and always returns the *same* handle for an encoding of the value v under the label l.

Furthermore, we say that $\mathcal{O}$ is a degree-$d(\cdot)$ ideal graded encoding oracle if for all $\lambda \in \mathbb{N}$, $g(1^\lambda, p, \vec{\mathbf{v}})$ returns 0 when $\mathsf{deg}(p) > d(\lambda)$.

Remark 2. *We remark that following [PST14b, Pas15], for simplicity of notation, our definition above do not directly allow players to create new encodings by adding and multiplying old ones (as in the definitions of [BR14, BGK⁺14]). As argued in [PST14a, Pas15], this restriction is without loss of generality for the purpose of constructing IO or VBB schemes. Roughly speaking, this is because the output encodings for addition and multiplication over old encodings can be easily emulated by giving "bogus" random handles. For this to work, it is important that a) the ideal graded encoding oracle allows adaptive encoding queries in Step 2 and b) we consider a model of* randomized *graded encodings (where multiple encodings of the same value are given fresh random handles). In such a model, the only case where the same handle will be returned twice is when they correspond to evaluating identical (formal) polynomials over the same set of handles; but, this behavior can be efficiently emulated.*

Moreover, Applebaum and Brakerski [AB15] show that even if the graded encodings are deterministic, that is, there is a unique encoding for every ring element. The interface of creating new encodings can still be emulated efficiently as long as the IO scheme satisfies a property called strong algebraic security *[AB15]. Our construction indeed satisfy this property (see Lemma 7), and hence is also secure in unique encoding models.*

5.4 IO in Oracle Model

We adapt the definition of IO in the plain model (Definition 17) to the oracle model, similar to the definitions of VBB obfuscation in oracle models in [CKP15, Pas15, MMN15]. For convenience, we will write an IO scheme as consisting of an obfuscator $i\mathcal{O}$, and additionally an evaluator Eval which specifies how to execute an obfuscated program produced by $i\mathcal{O}$ in the oracle model. Furthermore, below we present an equivalent security definition which states the view of an adversary after receiving an obfuscated program and interacting with the oracle, can be simulated by a potentially unbounded simulator. It follows from standard argument that this definition is equivalent to the indistinguishability based definition. In the analysis of our IO construction later, we will prove its security by constructing an unbounded simulator.

Definition 17 (Indistinguishability Obfuscator ($i\mathcal{O}$) in an Oracle Model)**.** *A pair of* PPT *oracle machines* $(i\mathcal{O}, \mathsf{Eval})$ *is an indistinguishability obfuscator for a class of* deterministic *circuits* $\{\mathcal{C}_\lambda\}_{\lambda\in\mathbb{N}}$ *in the $\mathcal{O}$-oracle model, if the following conditions are satisfied:*

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

$$\Pr[\hat{C} \leftarrow i\mathcal{O}^{\mathcal{O}_\lambda}(1^\lambda, C) \ : \ C(x) = \mathsf{Eval}^{\mathcal{O}_\lambda}(\hat{C})] = 1$$

μ-Unbounded Simulation Security: *For every polynomial-sized adversary $\mathcal{A}$, there exists a randomized (potentially unbounded) simulator* Sim *such that for every $\lambda \in \mathbb{N}$ and $C \in \mathcal{C}_\lambda$,*

$$\left|\Pr[\mathcal{A}^{\mathcal{O}_\lambda}(i\mathcal{O}^{\mathcal{O}_\lambda}(1^\lambda, C)) = 1] - \Pr[\mathsf{Sim}^C(1^\lambda) = 1]\right| \leq \mu(\lambda) \ .$$

6 IO for Special-Purpose Circuits in Ideal Model

In this section, we construct IO for our special-purpose circuits in ideal graded encoding model.

6.1 Overview

Our goal is to construct IO for $\{\{\mathcal{P}_\lambda^{T,n}\}\}$ with universal efficiency in constant degree ideal graded encoding model. Constructions of IO for NC^1 in the literature follow two approaches: Either obfuscate the branching programs of circuits [BR14, BGK+14, PST14a, GLSW14] or directly obfuscate circuits [AGIS14, Zim15, AB15]. The first approach seems to inherently require high-degree graded encodings, since the evaluation of a branching program has degree proportional to its length. This limitation does not hold for the second approach, but known constructions still require polynomial degree. We base our construction on the construction of IO for NC^1 by Applebaum and Brakerski [AB15] (shorthand AB-IO) in composite order ideal graded encoding model, and use new ideas to reduce the degree of graded encodings.

Review of Applebaum-Brakerski IO Scheme: Let P be a program with universal arithmetic circuit $U(x, P)$. Consider the following simple idea of encoding every bit of P and both values 0 and 1 for each input bit $i \in [n]$, that is, $\hat{P} = \{[b]_{\mathbf{v}_{i,b}}\}_{i\in[n],b\in\{0,1\}}, \{[P_i]_{\mathbf{v}_{i+n}}\}_{i\in[m]}$. Then, given an input x, an evaluator can simply pick the encodings $\{[x_i]_{\mathbf{v}_{i,x_i}}\}_{i\in[n]}$, and homomorphically evaluate U on encodings of (bits of) x and P to obtain an encoding of $U(x, P)$, which can then be learned by zero-testing. This simple idea does not go far. We mention several key issues and their solutions.

1. To prevent an adversary from using inconsistent values for the same input bit at different steps of the evaluation, AB-IO follows the standard solution of "straddling sets" [BGK+14], and uses a set of special levels, so that, if both $Z_{i,0} = [0]_{\mathbf{v}_{i,0}}$ and $Z_{i,1} = [1]_{\mathbf{v}_{i,1}}$ for some input bit i are used, the resulting encoding never reaches the zero testing level $\mathbf{v}_{zt}$. To see this, consider a simplified example: Set $\mathbf{v}_{i,0} = (1, 0, 1)$ and $\mathbf{v}_{i,1} = (0, 1, 1)$, and provide two additional encodings $\hat{Z}_{i,b}$ of random values under levels $\hat{\mathbf{v}}_{i,0} = (0, d, 0)$ and $\mathbf{v}_{i,1} = (d, 0, 0)$; the only way to reach level (d, d, d) is to use $Z_{i,b}$ consistently, followed by multiplication with $\hat{Z}_{i,b}$. Note that doing this for every input already requires degree n multiplication.

2. Graded encodings only support addition in the same levels. Since different input and program bits are encoded under different levels, homomorphic evaluation of U cannot be done. To resolve this, AB-IO uses El-Gamal encoding, under which a value w is represented as $(r, rw) \overset{\$}{\leftarrow} \mathsf{EG}(w)$ with a random r. Encodings of El-Gamal encodings of w_1 and w_2, $(R_1 = [r_1]_{\mathbf{v}_1}, Z_1 = [r_1w_1]_{\mathbf{v}_1})$ and $(R_2 = [r_2]_{\mathbf{v}_2}, Z_2 = [r_2w_2]_{\mathbf{v}_2})$ can be "added" using an addition gadget $\oplus$ that does $(R_1R_2 = [r_1r_2]_{\mathbf{v}_1+\mathbf{v}_2}, Z_1R_2 + Z_2R_1 = [r_1r_2(w_1 + w_2)]_{\mathbf{v}_1+\mathbf{v}_2})$, even if they are under different levels. Note that the new gadget, however, turns every addition in U into multiplications (and additions) in the homomorphic evaluation, which now has much higher degree, up to 2^{depth}, than U.

3. Point 1 ensures that an adversary must use an input x consistently, but, it can still deviate from evaluating U. AB-IO uses an information theoretic authentication method to prevent this. It samples a random value y_i for each input wire, and computes $\bar{y} = U(y_1, \cdots, y_{n+m})$. The idea is to use the structure of the composite order ring to "bind" the program and input bits with their corresponding y values, for example, instead of encoding $\mathsf{EG}(P_i)$, encode $\mathsf{EG}(w_{n+i})$ where $w_{n+i} = (P_i, y_{n+i})$. Therefore, whichever computation the adversary performs over x and P, the same is performed over $y_1, \cdots, y_{n+m}$. An honest evaluation yields encodings of $\mathsf{EG}((U(x, P), \bar{y}))$. By additionally releasing encodings of $\mathsf{EG}((1, \bar{y}))$, the output $U(x, P)$ can be learned by first subtracting the encodings and zero-test. Moreover, deviating from computing U leads to encodings of $\mathsf{EG}(Y(x, P), Y(y_1, \cdots, y_{n+m}))$ with some $Y \neq U$, and the value $Y(y_1, \cdots, y_{n+m})$ cannot be eliminated to allow zero-testing $Y(x, P)$, which hence remains hidden.

Due to Point 1 and 2, AB-IO requires the graded encodings to support degree-($n2^{depth}$) computations.

Towards Using Constant-Degree Graded Encodings, we modify AB-IO as follows:

1. We use the same method as AB-IO to prevent an adversary from using inconsistent input values, but we cannot afford to do that for every input bit. Instead, recall that the domain of our special purpose circuits is Σ^c, where Σ has size λ. We view each symbol $x^1, \cdots, x^c$ (though described as a λ-bit string) as a "single input", and apply the straddling sets of AB-IO for each input symbol. (Ignore the El-Gamal encoding and the y-values temporarily.) For the i^{th} symbol, release for every possible value $s \in \Sigma$, encoding $Z^i_s = [s]_{\mathbf{v}^i_s}$, and $\hat{Z}^i_s$ of a random value under set $\hat{\mathbf{v}}^i_s$. Consider a simplified example: Set $\mathbf{v}^i_s = (0 \cdots 0, 1, 0 \cdots 0, 1)$ with 1 at position s and $\lambda + 1$, and $\hat{\mathbf{v}}^i_s = (d \cdots d, 0, d \cdots d, 0)$ correspondingly. (As in Point 1 above,) the only way to reach $(d, \cdots, d)$ is using Z^i_s for some s consistently followed by a multiplication with $\hat{Z}^i_s$. The actual encoding is more complicated as s is described as a λ-bit string $s_1, \cdots, s_\lambda$, and each bit needs to be encoded separately $\vec{Z}^i_s = \{[s_j]_{\mathbf{v}^i_s}\}_j$.

2. Informally speaking, the addition gadget $\oplus$ of AB-IO turns addition over encodings under different levels into multiplication; to reduce the degree of homomorphic evaluation, we want to have as many additions under the same levels as possible. In particular, encodings of form $(R_1 = [r]_{\mathbf{v}}, Z_1 = [rw_1]_{\mathbf{v}})$ and $(R_2 = [r]_{\mathbf{v}}, Z_2 = [rw_2]_{\mathbf{v}})$ can be directly "added" $(R_1 = [r]_{\mathbf{v}}, Z_1 + Z_2 = [r(w_1 + w_2)]_{\mathbf{v}})$—we call this the constrained addition gadget $\tilde{\oplus}$. Fortunately, thanks to the special domain Σ^c, encodings for different bits of an input symbol $\vec{Z}^i_s$ have the same level $\mathbf{v}^i_s$. To allow for using $\tilde{\oplus}$, we further let their El-Gamal encodings share the same randomness r^i_s, that is, $R^i_s = [r^i_s]_{\mathbf{v}^i_s}$ and $\vec{Z}^i_s = \{[r^i_s s_j]_{\mathbf{v}^i_s}\}_j$. Now addition of different bits in the same input symbol can be performed using only homomorphic addition.

   More generally, we assign "types" to input wires—all wires describing P have one type, and these describing x^i for each i has another. Encodings for input wires of the same type share the same level and El-Gamal randomness, and can be added using $\tilde{\oplus}$ for "free", whereas addition across different types is done using $\oplus$ as in AB-IO, involving homomorphic multiplication. We further assign types to all wires in U recursively: When the incoming wires of an addition gate in U have the same types, $\tilde{\oplus}$ can be applied and its outgoing wire keeps the same type; in all other cases, homomorphic multiplication is required, and the types of the incoming wires add up. Careful examination reveals that the degree of homomorphic evaluation is proportional to the 1-norm of the output wire type, which we call the *type-degree of* U.

Combining the above ideas, we obtain a construction of IO for general circuit class in ideal model where the degree of the graded encodings is $O(td + c)$, proportional to the type degree td and the number of input type c of the circuit class; we say such a construction is *type degree preserving.*

For certain circuits, their type-degrees are much smaller than 2^{depth}. For example, our special purpose circuits, instantiated with a constant-degree PRG, have a constant type degree td, and hence constant degree graded encodings suffice. More generally, when PRG has degree $d(\lambda)$, the type degree of the special purpose circuits is polynomial in $d(\lambda)$.

Our actual IO scheme is more complicated than sketched above due to 1) it is based on the robust obfuscator in [AB15] as opposed to the simple obfuscator described above; like the robust obfuscator of [AB15], our IO scheme has the property that a generic attacker can only generate encodings of 0 at the zero-testing level. Such a construction can work with graded encoding schemes with unique encodings and seems to be more secure in face of zeroizing attacks on graded encodings.

In particular, [CGH+15] showed that a simplified version of the simple obfuscator of [AB15] can be attacked. 2) Our IO scheme directly obfuscates non-Boolean circuits. Previous constructions of IO for NC^1 considers only Boolean circuits; this is w.l.o.g. as a NC^1 circuit C can be turned into a Boolean one $\bar{C}(x, i) = C(x)_i$, still in NC^1. But, when aiming at type-degree preserving constructions of IO, we cannot use this trick, as $\bar{C}$ may have much higher type degree than C.

6.2 Type-Degree of Arithmetic Circuits

Definition 18 (The Type Function and Type Degree)**.** *Let Σ be any alphabet where every symbol in Σ is represented as a binary string of length $\ell \in \mathbb{N}$. Let $U(\star, \star)$ be an arithmetic circuit over domain $\Sigma^c \times \{0,1\}^K$ with some $K, c \in \mathbb{N}$. We say that U has c input-types and assign every wire w in U with a* type $\mathbf{t}_w \in \mathbb{N}^{c+1}$ *through the following recursively defined function* $\mathbf{t}_w = \mathsf{type}(U, w)$.

Base Case: *If w is the i^{th} input wire,*

- *For every $k \in [c]$, if $i \in [(k-1)\ell + 1, k\ell]$ (i.e., w describes the k^{th} symbol in the first input), assign type $\mathbf{t}_w = \mathbf{e}_{k+1}$.*
- *If $i \in [c\ell + 1, c\ell + K]$ (i.e., w describes the second input), assign type $\mathbf{t}_w = \mathbf{e}_1$.*

Recursion: *If w is the output wire of gate g with input wires u_0, u_1 of type $\mathbf{t}_{u_b} = \mathsf{type}(U, u_b)$ for $b \in \{0,1\}$,*

- *if g is an addition gate and $\mathbf{t}_{u_0} = \mathbf{t}_{u_1}$, assign type $\mathbf{t}_w = \mathbf{t}_{u_0}$;*
- *otherwise (i.e., g is a multiplication gate or $\mathbf{t}_{u_0} \neq \mathbf{t}_{u_1}$), assign type $\mathbf{t}_w = \mathbf{t}_{u_0} + \mathbf{t}_{u_1}$.*

Given the type function, we now define the quantity, *type-degree*, for arithmetic circuits, which is basically the *maximal 1-norm* of types of all wires in the circuit. Furthermore, when considering a class of circuits, we define its type-degree, by that of its universal arithmetic circuits.

Definition 19 (Type Degree)**.** *We define the type degree of the following objects:*

- *The type degree of a wire w in an arithmetic circuit U is $\mathsf{tdeg}(U, w) = |\mathsf{type}(U, w)|_1$.*
- *The type degree of an arithmetic U is $\mathsf{tdeg}(U) = \max_{w \in U}(\mathsf{tdeg}(U, w))$.*
- *We say that an ensemble of circuit families $\{\mathcal{P}_\lambda\}$ has $c(\star)$ types and type-degree $\mathsf{tdeg}(\star)$ if it has universal arithmetic circuits $\{U_\lambda\}$, where U_λ has $c(\lambda)$ input-types and type degree $\mathsf{tdeg}(\lambda)$.*

The following facts help understanding the notion of type degree. The first one establishes that type degree is an intermediate quantity that always sits between the (normal) degree of an arithmetic circuit and 2^d where d is the depth of the circuit.

Fact 1. *The type-degree of any arithmetic circuit U satisfies that $\deg(U) \leq \mathsf{tdeg}(U) \leq 2^d$, where d is the depth of U.*

Proof. The above fact can be established via induction over the depth of circuits. In the base case where $d = 0$, the input wires have type-degree 1, equal to both their degree and 2^0.

Suppose that the fact holds for all depth d circuits. Given a depth $d+1$ circuit U (for simplicity, U is Boolean), consider two cases: 1) If the final gate g is a multiplication gate, then the type degree of the output wire γ is the sum of that of the input wires u_0, u_1 of g, since (by the induction hypothesis) $\deg(C, u_b) \leq \mathsf{tdeg}(C, u_b) \leq 2^d$, we have that $\deg(C, \gamma) = \Sigma_b \deg(C, u_b) \leq \mathsf{tdeg}(C, \gamma) =$

$\Sigma_b \mathsf{tdeg}(C, u_b) \leq \Sigma_b 2^d = 2^{d+1}$; induction statement holds. 2) If the final gate γ is an addition gate, the type degree of γ is either the same as that of u_0, u_1 if they are of the same type, or is their sum. Since the degree of γ always stays the same as the degree of the input wires, it is easy to see that the induction statement holds again. □

The above fact is concerned with arithmetic circuits consisting of only fan-in 2 multiplication/addition gates. In particular, it shows that such arithmetic circuits with constant depth also has constant type degree. However, in general, arithmetic circuits with constant *degree* do not have constant type degree. Below, we slightly extend the above fact to constant-degree circuits with certain special structure. Later, we will show that our special-purpose circuits in fact have this special structure and hence have constant type degree.

Fact 2. *Let $U(\star, \star)$ be an arithmetic circuit consisting of fan-in 2 multiplication/addition gates and* additionally *unbounded fan-in addition gate (all with unbounded fan-out). If U satisfies the following property,*

- *for every unbounded fan-in addition gate, its input wires have the same type,*

Then the type degree of U is bounded by $\mathsf{tdeg}(U) \leq 2^d$, where d is the depth of U.

This fact essentially follows from the same proof as Fact 1. This is because every unbounded fan-in addition have the same input types and hence its output has the same type; informally speaking, the unbounded fan-in additions do not increase the type degree. Therefore, the type degree of such a circuit follows Fact 1 as if the unbounded fan-in additions were not there.

Recall that Proposition 6 shows that our special-purpose circuits can be implemented in constant degree. We now establish a stronger statement that they can in fact be implemented with constant type-degree.

Lemma 5 (The Special-Purpose Circuits Have Constant Type-Degree)**.** *The class of special-purpose circuits $\{\mathcal{P}_\lambda^{T,n}\}$ has universal arithmetic circuits $\{U_\lambda\}$ of constant $c^{T,n}$ input-types, constant type degree $\mathsf{tdeg}^{T,n}$, and size $u(1^\lambda, n, \log T)$, for a universal polynomial u independent of T, n.*

Proof. Fix any polynomials $T(\star)$ and $n(\star)$. The class $\{\mathcal{P}_\lambda^{T,n}\}$ in Figure 4 consists of special purpose circuits of the form $P[\lambda, T, n, \star_1](\star_2)$ (with $T = T(\lambda)$ and $n = n(\lambda)$), where $\star_1$ is for the rest of the constants (including $mpk, i^*, K, m_<, \hat{\Pi}, m_>$, where i^* is represented differently) and $\star_2$ is for the input $i \overset{\Sigma}{=} (\mathbf{e}_{i_1}, \cdots \mathbf{e}_{i_c})$ represented under alphabet $\Sigma = \{\mathbf{e}_0, \cdots, \mathbf{e}_\lambda\}$. Therefore, $U(\star_2, \star_1) = P[\lambda, T, n, \star_1](\star_2)$ can be viewed as a universal circuit of $\mathcal{P}_\lambda^{T,n}$, with domain $\Sigma^c \times \{0,1\}^K$. Hence, it has a constant number of input types. Moreover, it follows from the analysis of Proposition 6 that P has constant degree and size $u(\lambda, n, \log T)$ for some universal polynomial. Therefore, towards the lemma, it remains to show that P also has constant type degree.

Examine P in Figure 4. It has the structure that Step 1 evaluating the PRF F and Step 2 performing comparison $\mathsf{Compare}$ can be executed in parallel; their respective outputs $R_i || R'_i$ and $b_{i,<}, b_{i,=}, b_{i,>}$ are then fed into Step 3 and 4 to produce the final output. Observe that Step 3 and 4 can be implemented in NC^0 since RE is in NC^0. Thus, if the type degrees of (the wires corresponding to) $R_i || R'_i$ and $b_{i,<}, b_{i,=}, b_{i,>}$ are bounded by a constant t_{max}, then the type degree of the entire output is bounded by $2^c t_{max}$, where c is the constant depth of Step 3 and 4. Therefore, it suffices to prove the following two claims.

Claim 3. *All outputs of* $\mathsf{Compare}$ *in P have constant type degrees.*

We prove a stronger parameterized claim for the type degrees of the outputs of the PPRF F, w.r.t. the degree of PRG (not the type degree of PRG).

Claim 4. *If* PRG *had degree* $d(\lambda)$*, then all outputs of* F *in* P *have type degrees* $\mathrm{poly}(d(\lambda))$*.*

When PRG has constant degree, it follows immediately from the claim that the outputs of F in P all have constant type degrees. (We prove the above parameterized claim in order to reveal the general relation between the degree of PRG and the type degree of P, as we discuss later.)

In summary, the type degree of P is bounded by $2^c t_{max}$, where both c and t_{max} are constants. Therefore, P has constant type degree. Below we prove the above two claims.

Proof of 3. We use Fact 2 to show that outputs of $\mathsf{Compare}$ in P have constant type degrees. First observe that $\mathsf{Compare}$ can be implemented by a constant-depth arithmetic circuit consisting of fan-in 2 multiplication/addition gates and *additionally* unbounded fan-in addition gate. Examine the subroutines $\mathsf{Compare}$ in Figures 2. The inner product opeartions in Step 1 can be implemented using unbounded fan-in addition and fan-in 2 multiplication in depth 2; the rest Step 2-3 can be implemented using only fan-in 2 multiplication/addition in constant depth.

Therefore, by Fact 2, it suffices to show that for every inner product operation, the (unbounded fan-in) addition for computing it have the same input types. By definition of the type function, all input wires describing the constants (including $mpk, K, m_<, \hat{\Pi}, m_>$ and all values related to i^*) is assigned with type $\mathbf{e}_1 \in \mathbb{N}^{c+1}$, and input wires for $\mathbf{e}_{\mathsf{i}_k}$ with $k \in [c]$ is assigned with type $\mathbf{e}_{k+1} \in \mathbb{N}^{c+1}$. Then,

- In Step 1 of $\mathsf{Compare}$, inner products are used to perform block-wise comparisons. Recall that for every $k \in [c]$ and $\star \in \{=, <, >\}$, flag $b^k_{i,\star}$ indicates whether i_k is equal to, smaller than, or bigger than i^*_k, and is computed as $b^k_{i,\star} = \mathbf{e}_{\mathsf{i}_k} \cdot \mathbf{e}_{\star\ \mathsf{i}^*_k}$, where the first vector $\mathbf{e}_{\mathsf{i}_k}$ is a part of the input and has type $\mathbf{t}_1 = \mathbf{e}_{k+1}$, and the second vector $\mathbf{e}_{\star\ \mathsf{i}^*_k}$ is a part of the hardwired constants related to i^* and has type $\mathbf{t}_2 = \mathbf{e}_1$. Therefore, addition gates in this inner product operates over input wires with the same type $\mathbf{t}_1 + \mathbf{t}_2$.

□

Proof of Claim 4. Examine the structure of F in Figure 3. Step 1-2 producing choice bits d^k_i, and Step 3 producing strings K^k_c (both for $k \in [c]$) can be executed in parallel, whose outputs are then fed to Step 4 to produce the output. Since Step 4 is in NC^0, as argued before, the type degree of the circuit is bounded by $2^c t_{max}$, where c is the depth of Step 4 and t_{max} is the maximal type degree of the outputs of Step 1-2 and Step 3.

Observe that Step 1-2 have similar structure as the function $\mathsf{Compare}$ – step 1 computes inner products, followed by step 2 computing some constant opeartions. Thus it follows from similar proof of Claim 3 that their output bits d^k_i have constant type degree.

We now analyze the type degree of K^k_c for any $k \in [c]$ produced by Step 3. We prove by induction. The induction hypothesis is that after each iteration $k \le j \le c$, bits in the intermediate value K^k_j have the same type $\mathbf{t}^k_j$ and the type degree $|\mathbf{t}^k_j|_1 \le (2d)^j$, where $d = d(\lambda)$ is the degree of the PRG.

- *Base case* $j = k$*:* $K^k_k = \vec{v}_k[\ne i^*_k] \cdot \mathbf{e}_{\mathsf{i}_k}$, where the first vector is a part of the hardwired constant $K(-i^*)$ and has type $\mathbf{e}_1$, while the second vector has type $\mathbf{e}_{k+1}$; hence, the addition in the inner product operates over bits of the same type $\mathbf{t}^k_k = \mathbf{e}_1 + \mathbf{e}_{k+1}$, and producing bits in K^k_k with the same type. It follows immediately that $|\mathbf{t}^k_k|_1 \le 2 < (2d)^k$.

- *Induction:* Suppose that iteration $j-1$ produces K^k_{j-1} of type $\mathbf{t}^k_{j-1}$ whose ℓ_1 norm is bounded by $(2d)^{j-1}$. In the next iteration j, K^k_j is computed as

$$\vec{w}_j = \mathsf{qPRG}(K^k_{j-1}) \ , \ K^k_j = \vec{w}_j \cdot \mathbf{e}_{\mathsf{i}_j} \ .$$

  Assume with loss of generality that qPRG is represented as an arithemtic circuit with alternating (fan-in 2) addition and multplication layers, and all output bits have the same degree d (otherwise, it is easy to raise the degree by multiplying with constant 1). Since all input bits K^k_{j-1} have the same type $\mathbf{t}^k_{j-1}$, a simple induction proof shows that the bits in $\vec{w}_j$ all have the same type $\mathbf{t} = 2^e\mathbf{t}^k_{j-1}$, where e is the number of multiplication layers in qPRG. Note that $2^e = d$ as every multiplication layer doubles the degree of qPRG. Therefore, the induction hypothesis holds for j: All bits K^k_j have the same type $\mathbf{t}^k_j = d\mathbf{t}^k_{j-1} + \mathbf{e}_{j+1}$, and its ℓ_1 norm is bounded by $d|\mathbf{t}^k_{j-1}| + 1 \leq (2d)^j$.

This concludes the induction, and shows that K^k_c for any $k \in [c]$ have type degree bounded by $(2d)^c = \text{poly}(d)$. By the argument above, the type degree of F is bounded by $\text{poly}(d)$. $\square$

$\square$

Remark 3 (Generalization to Arbitrary PRG)**.** *When considering general PRG with arbitrary degree $d(\lambda)$, we can generalize Lemma 5 to the following lemma:*

Lemma 6 (The Special-Purpose Circuits Have Constant Type-Degree)**.** *If the underlying PRG* PRG *has degree $d(\lambda)$, the class of special-purpose circuits $\{\mathcal{P}^{T,n}_\lambda\}$ has universal arithmetic circuits $\{U_\lambda\}$ of constant $c^{T,n}$ input-types, type degree* $\mathsf{tdeg}^{T,n}(\lambda) = \text{poly}(d(\lambda))$*, and size $u(1^\lambda, n, \log T)$, for a universal polynomial u independent of T, n.*

The only difference from Lemma 5 is that the type-degree of the special-purpose circuits is now $\text{poly}(d(\lambda))$*, as opposed to a constant. This follows from the same argument as in the proof of Lemma 5. In particular, the degrees of the special-purpose circuits P are bounded by $2^c t_{max}$, where c is the constant depth of Step 3 and 4 in P, and t_{max} is the maximal type degrees of the outputs of* $\mathsf{Compare}$ *and* F*. The former by Claim 3 is always a constant, whereas the latter by Claim 4 is* $\text{poly}(d(\lambda))$*. Thus, the overall type degrees of the special-purpose circuits are* $\text{poly}(d(\lambda))$*.*

6.3 Type-Degree Preserving Construction of IO

In this section, we show that if a class of circuits has c input-types and type-degree td, then there is an IO scheme for it in the ideal graded encoding oracle model, where the oracle has degree $d = O(td + c)$. In other works, IO for classes of circuits with small (say constant) number of input-types and type-degree can be constructed from low (correspondingly constant) degree ideal graded encoding oracle. We say that such a construction is *type-degree preserving.*

Theorem 6 (Type-Degree Preserving Construction of IO)**.** *There is a uniform machine $i\mathcal{O}(\star, \star, \star)$ and a universal polynomial p, such that, the following holds:*

For any class of circuits $\{\mathcal{P}_\lambda\}$ that has universal arithmetic circuits $\{U_\lambda\}$ with $c(\lambda)$ input-types, type degree $\mathsf{tdeg}(\lambda)$*, and size $S(\lambda)$, there is an ideal graded encoding oracle $\mathcal{O} = \{\mathcal{O}^{g,\tilde{\mathcal{R}}}_\lambda\}$ with degree $d(\lambda) = O(\mathsf{tdeg}(\lambda) + c(\lambda))$, such that, $i\mathcal{O}_\mathcal{P}(\star, \star) = i\mathcal{O}(U_\lambda, \star, \star)$ is a (sub-exponentially secure) indistinguishability obfuscator for $\{\mathcal{P}_\lambda\}$ in the $\mathcal{O}$-oracle model, with run time $p(1^\lambda, S(\lambda))$ (for sufficiently large λ).*

By the above theorem, we can obtain a family of IO schemes $\{i\mathcal{O}^{T,n}\}$ for the family $\{\{\mathcal{P}_\lambda^{T,n}\}\}$ with universal efficiency, relying only on constant-degree ideal graded encoding oracles.

Proposition 7. *There is a family of (sub-exponentially secure) IO schemes $\{i\mathcal{O}^{T,n}\}$ for the family of special-purpose circuit classes $\{\{\mathcal{P}_\lambda^{T,n}\}\}$ in Figure 4 with universal efficiency, such that, for every polynomials T, n, $i\mathcal{O}^{T,n}$ uses an ideal graded encoding oracle with a constant degree $d^{T,n}$.*

Proof. Lemma 5 states that for any polynomials T and n, our special purpose circuit class $\{\mathcal{P}_\lambda^{T,n}\}$ has universal arithmetic circuits $\{U_\lambda^{T,n}\}$ with constant number of input types $c^{T,n}$ and type-degree $td^{T,n}$, and polynomial size $u(\lambda, n, \log T)$ for some universal polynomial u. By Theorem 6, there is an ideal graded encoding oracle $\mathcal{O}^{T,n}$ of constant degree $d^{T,n}$, such that, $i\mathcal{O}(U_\lambda^{T,n}, 1^\lambda, \star)$ is an IO scheme for the circuit class $\{\mathcal{P}_\lambda^{T,n}\}$ in the $\mathcal{O}^{T,n}$-oracle model and runs in time $p(\lambda, |U_\lambda^{T,n}|)$.

We now construct a family of IO schemes $\{i\mathcal{O}^{T,n}\}$ for $\{\{\mathcal{P}_\lambda^{T,n}\}\}$, by setting

$$i\mathcal{O}^{T,n}(1^\lambda, \star) = i\mathcal{O}(U_\lambda^{T,n}, 1^\lambda, \star) \ .$$

Additionally, by the efficiency of $i\mathcal{O}$, the run-time of $i\mathcal{O}^{T,n}$ is

$$\mathsf{Time}_{i\mathcal{O}^{T,n}}(1^\lambda, P) = \mathsf{Time}_{i\mathcal{O}}(U_\lambda^{T,n}, 1^\lambda, P) \leq p(\lambda, |U_\lambda^{T,n}|) \ .$$

Since the size of the universal arithmetic circuit $U_\lambda^{T,n}$ is bounded by $u(\lambda, n(\lambda), \log T(\lambda))$, we have,

$$\mathsf{Time}_{i\mathcal{O}^{T,n}}(1^\lambda, P) \leq p(\lambda, |U_\lambda^{T,n}|) \leq p(\lambda, u(\lambda, n(\lambda), \log T(\lambda))) \ .$$

This shows that the family $\{i\mathcal{O}^{T,n}\}$ has universal efficiency. □

Remark 4 (Generalization to Arbitrary PRG:)**.** *We remark that it follows from Lemma 6 that when the PRG* PRG *underlying $\{\mathcal{P}_\lambda^{T,n}\}$ have degree $d(\lambda)$ (instead of a constant), the type degree of their universal arithmetic circuits is* $\mathrm{poly}(d(\lambda))$*. Therefore, by the type-degree preserving IO construction of Theorem 6, IO for $\{\mathcal{P}_\lambda^{T,n}\}$ exists in a degree* $\mathrm{poly}(d(\lambda))$ *ideal graded encoding model. This concludes the second bullet of Proposition 4 discussed in the introduction.*

Remark 5 (Discussion on Bootstrapping in the Oracle Model)**.** *It is natural to ask whether our bootstrapping technique (Proposition 5) can be applied to transform the above IO for special-purpose circuits in the oracle model to a full-fledged IO for* P/poly *also in the oracle model. However, the short answer is that the bootstrapping technique only works with constructions in the plain model. More specifically, recall that our bootstrapping proceeds in two steps—first obtaining a compact FE and then invoke the more refined theorem of [AJ15, BV15a] to obtain IO for* P/poly*. For the first step of constructing a compact FE* $\mathsf{FE}^{T,n}$ *for* NC^1 *circuits of size T and n input bits (in Section 3.3), if we plug in an IO scheme for $\{\mathcal{P}_\lambda^{T,n}\}$ in the ideal constant-degree GE model, we obtain a compact FE* $\mathsf{FE}^{T,n}$ *also in the ideal constant-degree GE model; in particular, in this scheme, the encryption algorithm activates and uses the encoding procedure* OEnc *of the oracle, and the decryption algorithm uses the evaluation procedure* OEval *of the oracle. However, the second step of bootstrapping from compact FE to IO for* P/poly *of [AJ15, BV15a] only works if the compact FE is in the plain model, since their transformation requires using the FE scheme to generate secret keys for the encryption algorithm of the scheme itself. In other words, it requires either the encryption algorithm of the FE scheme to be in the plain model, or the scheme can generate secret key for functions using oracles.*

6.4 Type Degree Preserving Construction of IO

Consider an arbitrary circuit class $\{\mathcal{P}_\lambda\}$ with universal circuits $\{U_\lambda\}$. The universal circuit $U = U_\lambda$ has the following parameters:

- degree $d = \deg(U)$,
- domain $\{0,1\}^m \times \Sigma^c$, meaning every circuit $P \in \mathcal{P}$ is described by a m-bit string, and every input $x \overset{\Sigma}{=} x^1, \cdots, x^c$ is represented under alphabet Σ, i.e., $x^k \in \Sigma$ for every $k \in [c]$,
- alphabet Σ with q symbols that can be enumerated under a canonical order,
- a set Γ of γ output wires, and denote by U_o the induced Boolean circuit with output wire o, by $\mathbf{t}_o = \mathsf{type}(U, o)$ the type of that wire, and d_o the degree.
- Let $M = \max_{o\in\Gamma}(|\mathbf{t}_o|_\infty)$.

Below, we first describe our IO scheme $i\mathcal{O}_{\mathcal{P}}(1^\lambda, \star) = i\mathcal{O}(U, 1^\lambda, \star)$, and its evaluation procedure Eval. For simplicity of notation, we suppress the subscript $\mathcal{P}$ and implicitly assume that $i\mathcal{O}$ knows about the universal circuit U below.

Encoding Levels: We specify the levels used in the IO construction, as well as the final zero-testing level $\mathbf{v}_{zt}$ in Figure 5. All levels are represented as a $(q+1) \times (c+2)$ matrix followed by a $\gamma \times 1$ vector over $\mathbb{N}$, where $q = |\Sigma|$ and $\gamma = |\Gamma|$. [3] Below we describe them column by column. Let $\mathbf{1}_{i_1,i_2,\cdots,i_p}$ denote a vector with ones at positions $i_1, i_2, \cdots, i_p$ and zeros everywhere else, $\mathbf{1}_{\neq i_1,i_2,\cdots,i_p}$ a vector with zeros at positions $i_1, i_2, \cdots, i_p$ and ones everywhere else, and $\mathbf{0}$ the all 0 vector. $a \otimes b$ denote the tensor product between the two vectors.

For convenience, we overload notations to let s and o also denote their indexes in the alphabet Σ and the set of output wires Γ (with range $[\Sigma]$ or $[\Gamma]$), which allows us to write for example $\mathbf{1}_s$, $\mathbf{1}_o$. We also do not explicitly mention the length of the columns in the matrices, which is implicitly assumed to be $q+1$, and the length of the vectors (following the matrices) are assumed to be γ.

The Obfuscator $i\mathcal{O}$: on input 1^λ and $P \in \mathcal{P}$, proceeds as follows:
Activation: Activate the GE Oracle $\mathcal{O}$ and receive a ring $\mathcal{R} \cong \mathcal{R}_1 \times \mathcal{R}_2, \cdots \times \mathcal{R}_{c+3}$.
Encoding: Invoke the encoding procedure OEnc of $\mathcal{O}$ to encode the following elements, under levels defined below.

1. Obtain encoding $Z^* = [w^*]_{\mathbf{v}^*}$, for $w^* = (1, 1, \rho^*_1, \cdots, \rho^*_{c+1})$ that is random on all but the first two subgroup, in which the elements are 1.
2. For $k \in [c]$, encode for the $k-1$th input symbol:
   - For every symbol $s \in \Sigma$, obtain encoding $R^k_s = [r^k_s]_{\mathbf{v}^k_s}$ for random element $r^k_s \overset{\$}{\leftarrow} \mathcal{R}$.
   - For every symbol $s \in \Sigma$, and every jth bit s_j, obtain encoding $Z^k_{s,j} = [r^k_s \cdot w^k_{s,j}]_{\mathbf{v}^k_s + \mathbf{v}^*}$ for element $w^k_{s,j} \cong (y^k_j, s_j, \rho^k_{s,j,1}, \cdots, \rho^k_{s,j,c+1})$ with $y^k_j \overset{\$}{\leftarrow} \mathcal{R}_1$ and $(\rho^k_{s,j,1}, \cdots, \rho^k_{s,j,c+1}) \overset{\$}{\leftarrow} \mathcal{R}_3 \times \cdots \mathcal{R}_{c+3}$.
3. For $k = c+1$, encode the program description:

[3] Equivalently, they are $(|\Sigma|+1) \times (c+2) + |\Gamma|$ long vectors; we choose the "matrix plus vector" representation for ease of exposition.

$$\mathbf{v}^* = (\mathbf{0}, \cdots, \mathbf{0}, \mathbf{1}_{q+1}), \mathbf{0} \qquad \mathbf{v}_s^k = (\mathbf{0}, \cdots, \mathbf{1}_s, \cdots, \mathbf{0}), \mathbf{0}$$

$$\mathbf{v}^* = \left[\begin{array}{ccccc|c} 0 & \cdots & 0 & \cdots & 0 & 0 \\ \vdots & & \vdots & & \vdots & \vdots \\ 0 & \cdots & 0 & \cdots & 0 & 0 \\ \vdots & & \vdots & & \vdots & \vdots \\ 0 & \cdots & 0 & \cdots & 0 & 0 \\ \hline 0 & \cdots & 0 & \cdots & 0 & 1 \end{array}\right], \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \qquad \mathbf{v}_s^k = \left[\begin{array}{ccccc|c} 0 & \cdots & 0 & \cdots & 0 & 0 \\ \vdots & & \vdots & & \vdots & \vdots \\ 0 & \cdots & 0 & \cdots & 0 & 0 \\ 0 & \cdots & 1 & \cdots & 0 & 0 \\ 0 & \cdots & 0 & \cdots & 0 & 0 \\ \vdots & & \vdots & & \vdots & \vdots \\ 0 & \cdots & 0 & \cdots & 0 & 0 \\ \hline 0 & \cdots & 0 & \cdots & 0 & 0 \end{array}\right], \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

$$\mathbf{v}^{c+1} = (\mathbf{0}, \cdots \mathbf{0}, \mathbf{1}_{\neq q+1}, \mathbf{0}), \mathbf{0}$$

$$\hat{\mathbf{v}}_{s,o}^k = (\mathbf{1}_{\neq q+1} \otimes (0 \cdot \cdot 0, \mathbf{t}_o[k], 0 \cdot \cdot 0) + \mathbf{1}_{q+1} \otimes (0 \cdot \cdot 0, 1, 0 \cdot \cdot 0)), \mathbf{1}_o$$

$$\mathbf{v}^{c+1} = \left[\begin{array}{ccccc|c} 0 & \cdots & 0 & \cdots & 1 & 0 \\ \vdots & & \vdots & & \vdots & \vdots \\ 0 & \cdots & 0 & \cdots & 1 & 0 \\ \vdots & & \vdots & & \vdots & \vdots \\ 0 & \cdots & 0 & \cdots & 1 & 0 \\ \hline 0 & \cdots & 0 & \cdots & 0 & 0 \end{array}\right], \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \qquad \hat{\mathbf{v}}_{s,o}^k = \left[\begin{array}{ccccc|c} 0 & \cdots & \mathbf{t}_o[k] & \cdots & 0 & 0 \\ \vdots & & \vdots & & \vdots & \vdots \\ 0 & \cdots & \mathbf{t}_o[k] & \cdots & 0 & 0 \\ 0 & \cdots & 0 & \cdots & 0 & 0 \\ 0 & \cdots & \mathbf{t}_o[k] & \cdots & 0 & 0 \\ \vdots & & \vdots & & \vdots & \vdots \\ 0 & \cdots & \mathbf{t}_o[k] & \cdots & 0 & 0 \\ \hline 0 & \cdots & 1 & \cdots & 0 & 0 \end{array}\right], \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

$$\hat{\mathbf{v}}_o = (\mathbf{1}_{\neq q+1} \otimes (M - \mathbf{t}_o[1] \cdots, M - \mathbf{t}_o[c+1],\ 0) + \mathbf{1}_{q+1} \otimes (0, \cdots, 0, 1, 0)),\ c \cdot \mathbf{1}_{\neq o}$$

$$\hat{\mathbf{v}}_o = \left[\begin{array}{ccccc|c} M - \mathbf{t}_o[1] & \cdots & M - \mathbf{t}_o[k] & \cdots & M - \mathbf{t}_o[c+1] & 0 \\ \vdots & & \vdots & & \vdots & \vdots \\ M - \mathbf{t}_o[1] & \cdots & M - \mathbf{t}_o[k] & \cdots & M - \mathbf{t}_o[c+1] & 0 \\ \vdots & & \vdots & & \vdots & \vdots \\ M - \mathbf{t}_o[1] & \cdots & M - \mathbf{t}_o[k] & \cdots & M - \mathbf{t}_o[c+1] & 0 \\ \hline 0 & \cdots & 0 & \cdots & 1 & 0 \end{array}\right], \begin{bmatrix} c \\ \vdots \\ c \\ 0 \\ c \\ \vdots \\ c \end{bmatrix}$$

$$\bar{\mathbf{v}}_o = (\mathbf{0}, \cdots, \mathbf{0}, \mathbf{1}_{\neq q+1}), \mathbf{0}$$

$$\mathbf{v}_{zt} = (\mathbf{1}_{\neq q+1} \otimes (M, \cdots, M, 1) + \mathbf{1}_{q+1} \otimes (1, \cdots, 1, D)), c \cdot \mathbf{1}$$

$$\bar{\mathbf{v}}_o = \left[\begin{array}{ccccc|c} 0 & \cdots & 0 & \cdots & 0 & 1 \\ \vdots & & \vdots & & \vdots & \vdots \\ 0 & \cdots & 0 & \cdots & 0 & 1 \\ \vdots & & \vdots & & \vdots & \vdots \\ 0 & \cdots & 0 & \cdots & 0 & 1 \\ \hline 0 & \cdots & 0 & \cdots & 0 & 0 \end{array}\right], \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \qquad \mathbf{v}_{zt} = \left[\begin{array}{ccccc|c} M & \cdots & M & \cdots & M & 1 \\ \vdots & & \vdots & & \vdots & \vdots \\ M & \cdots & M & \cdots & M & 1 \\ \vdots & & \vdots & & \vdots & \vdots \\ M & \cdots & M & \cdots & M & 1 \\ \hline 1 & \cdots & 1 & \cdots & 1 & D \end{array}\right], \begin{bmatrix} c \\ \vdots \\ c \\ c \\ c \\ \vdots \\ c \end{bmatrix}$$

Figure 5: Levels used in obfuscation. Matrices have dimension $(q+1) \times (c+2)$ and vectors have dimension γ. In the matrices of $\mathbf{v}_s^k$ and $\hat{\mathbf{v}}_{s,o}^k$, the k^{th} column is non-zero, the row corresponding to the index of s in $\mathbf{v}_s^k$ and $\hat{\mathbf{v}}_{s,o}^k$ is non-zero and zero respectively. In the vectors of $\hat{\mathbf{v}}_{s,o}^k$ and $\hat{\mathbf{v}}_o$, the position corresponding to the index of o is non-zero and zero respectively.

- Obtain encoding $R^k = [r^k]_{\mathbf{v}^k}$ for random element $r^k \xleftarrow{\$} \mathcal{R}$.
- For every $j \in [m]$, obtain encoding $Z^k_j = [r^k \cdot w^k_j]_{\mathbf{v}^k+\mathbf{v}^*}$ for element $w^k_j \cong (y^k_j, P_j, \rho^k_{j,1}, \cdots, \rho^k_{j,c+1})$ with $y^k_j \xleftarrow{\$} \mathcal{R}_1$, and $(\rho^k_{j,1}, \cdots, \rho^k_{j,c+1}) \xleftarrow{\$} \mathcal{R}_3 \times \cdots \mathcal{R}_{c+3}$.

4. For every output wire $o \in \Gamma$, encode elements for straddling purposes.
   - "Input consistency": For every $k \in [c]$ and every symbol $s \in \Sigma$, obtain encoding $\hat{R}^k_{s,o} = [\hat{r}^k_{s,o}]_{\hat{\mathbf{v}}^k_{s,o}}$ and $\hat{Z}^k_{s,o} = [\hat{r}^k_{s,o} \cdot \hat{w}^k]_{\hat{\mathbf{v}}^k_{s,o}+\mathbf{v}^*}$ for random elements $\hat{r}^k_{s,o} \xleftarrow{\$} \mathcal{R}$ and element $\hat{w}^k = (\hat{y}^k, \hat{\beta}^k, \hat{\rho}^k_1, \cdots, \hat{\rho}^k_{c+1})$ that is random in all subrings except the $k+2^{\text{th}}$, in which $\hat{\rho}^k_k = 0$.
   - "Output consistency": Obtain encoding $\hat{R}_o = [\hat{r}_o]_{\hat{\mathbf{v}}_o}$ and $\hat{Z}_o = [\hat{r}_o \cdot \hat{w}]_{\hat{\mathbf{v}}_o+\mathbf{v}^*}$ for random elements $\hat{r}_o \xleftarrow{\$} \mathcal{R}$ and element $\hat{w} = (\hat{y}, \hat{\beta}, \hat{\rho}_1, \cdots, \hat{\rho}_{c+1})$ that is random in all subrings except the $c+3^{\text{rd}}$, in which $\hat{\rho}_{c+1} = 0$.
5. For every output wire $o \in \Gamma$, encode an element for authentication purpose.
   - Obtain encoding $\bar{R}_o = [\bar{r}_o]_{\bar{\mathbf{v}}}$ and $\bar{Z}_o = [\bar{r}_o \cdot \bar{w}_o]_{\bar{\mathbf{v}}+D\mathbf{v}^*}$ for random element $\bar{r}_o \xleftarrow{\$} \mathcal{R}$, and

$$\bar{w}_o = \Big(\hat{w} \cdot \prod_{k\in[c]} \hat{w}^k\Big)\,(\bar{y}_o,\ 1,\ 0,\ \cdots, 0)\ , \text{ where}$$
$$\bar{y}_o = U_o\left(\{y^1_{x^1,j}\}, \cdots, \{y^{c+1}_{x^{c+1},j}\}, \{y^{c+1}_j\}\right)\ .$$

   where $D = d + c + 1$, and $d = \deg(U)$ is the degree of U.

Output: The obfuscated program consists of

$$\hat{P} = \Big(Z^*,\ \Big\{\Big(R^k_s, Z^k_{s,j}\Big)\Big\}_{k\in[c],s\in\Sigma,j\in[\ell]},\ \Big\{\Big(R^{c+1}, Z^{c+1}_j\Big)\Big\}_{j\in[m]},$$
$$\Big\{\Big(\hat{R}^k_{s,o}, \hat{Z}^k_{s,o}\Big)\Big\}_{k\in[c],s\in\Sigma,o\in\Gamma},\ \Big\{\Big(\hat{R}_o, \hat{Z}_o\Big)\Big\}_{o\in\Gamma},\ \Big\{\Big(\bar{R}_o, \bar{Z}_o\Big)\Big\}_{o\in\Gamma}\Big)\ .$$

The Evaluator Eval: on input an obfuscated program $\hat{P}$ and input $x = x^1, \cdots, x^c \in \Sigma^c$, compute $y_o = U_o(x, P)$ for every output wire $o \in \Gamma$ using the zero testing procedure OEval of $\mathcal{O}$ as follows.

We in-line the analysis of correctness in the description below.

1. Consider the following two arrays $\mathbf{A}_x$ and $\mathbf{B}_{x,o}$ of encodings, chosen from $\hat{P}$ according to $x^1, \cdots, x^c$, and output wire o:

$$\mathbf{A}_x = \Big(\Big\{\Big(R^k_{x^k}, Z^k_{x^k,j}\Big)\Big\}_{k\in[c],j\in[\ell]},\ \Big\{\Big(R^{c+1}, Z^{c+1}_j\Big)\Big\}_{j\in[m]}\Big)$$
$$\mathbf{B}_{x,o} = \Big(\Big\{\Big(\hat{R}^k_{x^k,o}, \hat{Z}^k_{x^k,o}\Big)\Big\}_{k\in[c]},\ \Big(\hat{R}_o, \hat{Z}_o\Big),\ \Big(\bar{R}_o,\ \bar{Z}_o\Big)\Big)\ .$$

2. Perform the following *mental* evaluation over encodings $(Z^*, \mathbf{A}_x)$, with the goal of emulating the computation of U_o. More specifically, we recursively associate every wire α in U_o with a pair of encodings (R_α, Z_α) of elements in El-Gamal form, that is, $R_\alpha = [r_\alpha]_{\mathbf{v}_\alpha}$, $Z_\alpha = [r_\alpha \cdot w_\alpha]_{\mathbf{v}_\alpha+d_\alpha\mathbf{v}^*}$ as follows:

Input: Encoding Z^* and two pairs of encoding $(R_\alpha = [r_\alpha]_{\mathbf{v}_\alpha}, Z_\alpha = [r_\alpha \cdot w_\alpha]_{\mathbf{v}_\alpha + d_\alpha \mathbf{v}^*})$ and $(R_\beta = [r_\beta]_{\mathbf{v}_\beta}, Z_\beta = [r_\beta \cdot w_\beta]_{\mathbf{v}_\beta + d_\beta \mathbf{v}^*})$. Permute the operands to ensure that $\delta = d_\beta - d_\alpha \geq 0$.
Output: A pair of encoding $(R_\sigma = [r_\sigma]_{\mathbf{v}_\sigma}, Z_\sigma = [r_\sigma \cdot w_\sigma]_{\mathbf{v}_\sigma + d_\sigma \mathbf{v}^*})$.

Gadget Multiplication $\otimes$

- $R_\sigma = R_\alpha \times R_\beta$ and $Z_\sigma = Z_\alpha \times Z_\beta$.
  It satisfies that $r_\sigma = r_\alpha \cdot r_\beta$, $w_\sigma = w_\alpha \cdot w_\beta$, $\mathbf{v}_\sigma = \mathbf{v}_\alpha + \mathbf{v}_\beta$, and $d_\sigma = d_\alpha + d_\beta$.

Gadget Addition $\oplus$ / Subtraction $\ominus$

- $R_\sigma = R_\alpha \times R_\beta$ and $Z_\sigma = Z_\alpha \times R_\beta \times (Z^*)^\delta \; +/- \; Z_\beta \times R_\beta$.
  It satisfies that $r_\sigma = r_\alpha \cdot r_\beta$, $w_\sigma = w_\alpha \cdot (w^*)^\delta \; +/- \; w_\beta$, $\mathbf{v}_\sigma = \mathbf{v}_\alpha + \mathbf{v}_\beta$, and $d_\sigma = d_\beta$.

Constrained Gadget Addition $\tilde{\oplus}$ / Subtraction $\tilde{\ominus}$

- **Constraint:** $r = r_\alpha = r_\beta$ and $\mathbf{v} = \mathbf{v}_\alpha = \mathbf{v}_\beta$.
- $R_\sigma = R_\alpha$ and $Z_\sigma = Z_\alpha \times (Z^*)^\delta \; +/- \; Z_\beta$.
  It satisfies that $r_\sigma = r$, $w_\sigma = w_\alpha \cdot (w^*)^\delta \; +/- \; w_\beta$, $\mathbf{v}_\sigma = \mathbf{v}$, and $d_\sigma = d_\beta$.

Remark: Since $w^*[\![b]\!] = 1$ for $b \in \{1, 2\}$, in $\otimes / \oplus, \tilde{\oplus} \; / \; \ominus, \tilde{\ominus}$, $w_\sigma[\![b]\!] = w_\alpha[\![b]\!] \times/+/- \; w_\beta[\![b]\!]$. Furthermore, d_σ grows exactly as how the degree of a polynomial grows: It adds up in multiplication $d_\sigma = d_\alpha + d_\beta$ and takes the larger value in addition/subtraction $d_\sigma = d_\beta$.

Figure 6: Gadgets used to replace simple addition and multiplication for computing over El-Gamal encoding

In the base case, the input wires correspond exactly to pairs of encodings in $\mathbf{A}_x$, that is, the j^{th} input wire of x^k is associated with pair $(R^k_{x^k}, Z^k_{x^k,j})$ and that of P is associated with (R^{c+1}, Z^{c+1}_j).

In the induction case, for every gate g in U with input wires α, β and output wire σ of types $\mathbf{t}_\alpha, \mathbf{t}_\beta, \mathbf{t}_\sigma$ respectively, apply the appropriate gadget as described in Figure 6, over encodings Z^* and $(R_\alpha, Z_\alpha), (R_\beta, Z_\beta)$. (For simplicity of notation, we ignore Z^* from the inputs to the gadgets below.)

- If g is a $\times$ gate, compute $(R_\sigma, Z_\sigma) = (R_\alpha, Z_\alpha) \otimes (R_\beta, Z_\beta)$.
- If g is an $+/-$ gate and $\mathbf{t}_\alpha \neq \mathbf{t}_\beta$, compute $(R_\sigma, Z_\sigma) = (R_\alpha, Z_\alpha) \oplus / \ominus (R_\beta, Z_\beta)$.
- If g is an $+/-$ gate and $\mathbf{t}_\alpha = \mathbf{t}_\beta$, compute $(R_\sigma, Z_\sigma) = (R_\alpha, Z_\alpha) \tilde{\oplus} / \tilde{\ominus} (R_\beta, Z_\beta)$.

Let $U'_o(Z^*, \mathbf{A}_x)$ be the arithmetic circuit evaluated, and (R_o, Z_o) the pair of encodings associated with the output wire o.

Correctness Analysis:

The encodings (R_o, Z_o) has form $R_o = [r_o]_{\mathbf{v}_o}$, $Z_o = [r_o \cdot w_o]_{\mathbf{v}_o + d_o \mathbf{v}^*}$ and

$$\text{where } w_o = \left(U_o\left(\{y^1_j\}_{j \in [\ell]}, \; \cdots \{y^c_j\}_{j \in [\ell]}, \; \{y^{c+1}_j\}_{j \in [m]} \right), \; U_o(x^1, \cdots, x^c, P), \; \star, \cdots, \star \right),$$
$$\text{and } \mathbf{v}_o = \left(\mathbf{t}_o[c] \cdot \mathbf{1}_{x^1}, \; \cdots, \; \mathbf{t}_o[c] \cdot \mathbf{1}_{x^c}, \; \mathbf{t}_o[c+1] \cdot \mathbf{1}_{\neq q+1}, \; \mathbf{0} \right), \; \mathbf{0} \; ,$$
$$\text{and } d_o = \deg(U_o).$$

In the above, the values denoted by $\star$ do not matter for correctness, and hence are not mentioned explicitly.

---

3. Perform the following *mental* evaluation over encodings $(Z^*, (R_o, Z_o), \mathbf{B}_{x,o})$

   - For "input consistency" purposes:

   $$R'_o = \Big(\prod_k \hat{R}^k_{x^k,o}\Big) R_o \ , \qquad Z'_o = \Big(\prod_k \hat{Z}^k_{x^k,o}\Big) Z_o \ .$$

   - For "output consistency" purposes: $R''_o = \hat{R}_o R'_o$, $Z''_o = \hat{Z}_o Z'_o$.
   - For "authentication" purpose: $(\tilde{R}_o, \tilde{Z}_o) = (R'_o, Z'_o) \ominus (\bar{R}_o, \bar{Z}_o)$.

   Let $\tilde{Z}_o = C_o((R_o, Z_o), \mathbf{B}_{x,o}))$ be the arithmetic circuit evaluated, which outputs *only* $\tilde{Z}_o$.

   Correctness Analysis:

   - *The encodings (R'_o, Z'_o) satisfy $R'_o = [r'_o]_{\mathbf{v}'_o}$, $Z'_o = [r'_o \cdot w'_o]_{\mathbf{v}'_o + (d_o + c)\mathbf{v}^*}$ and,*

   $$w'_o = \big(\prod \hat{w}^k\big) \Big(U_o \Big(\{y^1_j\}, \ \cdots \{y^c_j\}, \ \{y^{c+1}_j\}\Big) \ , \ U_o(x^1, \cdots, x^c, P), \ 0 \cdots 0, \star\Big)$$
   $$\mathbf{v}'_o = \left(\mathbf{1}_{\neq q+1} \otimes (\mathbf{t}_o[1], \ \cdots \ , \ \mathbf{t}_o[c+1], \ 0) + \mathbf{1}_{q+1}(1, \cdots, 1, 0, 0)\right), \ c \cdot \mathbf{1}_o$$

   - *The encodings (R''_o, Z''_o) satisfy $R''_o = [r''_o]_{\mathbf{v}''_o}$, $Z''_o = [r''_o \cdot w''_o]_{\mathbf{v}''_o + (d_o + c + 1)\mathbf{v}^*}$,*

   $$w''_o = \hat{w}\big(\prod \hat{w}^k\big) \Big(U_o \Big(\{y^1_j\}, \ \cdots \{y^c_j\}, \ \{y^{c+1}_j\}\Big) \ , \ U_o(x^1, \cdots, x^c, P), \ 0 \cdots 0, 0\Big)$$
   $$\mathbf{v}''_o = \left(\mathbf{1}_{\neq q+1} \otimes (M, \ \cdots \ , \ M, \ 0) + \mathbf{1}_{q+1} \otimes (1, \cdots, 1, 1, 0)\right), \ c \cdot \mathbf{1}$$

   - *The encodings $(\tilde{R}_o, \tilde{Z}_o)$ satisfy $\tilde{R}_o = [\tilde{r}_o]_{\tilde{\mathbf{v}}}$, $\tilde{Z}_o = [\tilde{r}_o \cdot \tilde{w}_o]_{\tilde{\mathbf{v}} + D\mathbf{v}^*}$ and,*

   $$\tilde{w}_o = (w^*)^{d - d_o} w''_o - \bar{w}_o = \hat{w}\big(\prod \hat{w}^k\big) \left(0, \ U_o(x^1, \cdots, x^c, P) - 1, \ 0 \cdots 0, 0\right)$$
   $$\tilde{\mathbf{v}} = \left(\mathbf{1}_{\neq q+1} \otimes (M, \cdots, M, 1) + \mathbf{1}_{q+1} \otimes (1, \cdots, 1, 0)\right), c \cdot \mathbf{1}$$

   Iff $U_o(x^1, \cdots, x^c, P) = 1$, $\tilde{w}_o = 0$ and $\tilde{Z}_o$ is an encoding of 0 under $\mathbf{v}_{zt} = \tilde{\mathbf{v}} + D\mathbf{v}^*$.

   ---

4. Call the zero-testing procedure OEval of $\mathcal{O}$ with the polynomial

   $$\tilde{Z}_o = U''_o(Z^*, \mathbf{A}_x, \mathbf{B}_{x,o}) = C_o(Z^*, U'_o(Z^*, \mathbf{A}_x), \mathbf{B}_{x,o}) \ .$$

   Output $y_o = 1$ iff the zero-test returns 1.

As analyzed in the box above, in an honest evaluation, $\tilde{Z}_o$ is an encoding of 0 under level $\mathbf{v}_{zt}$ iff $y_o = 1$; hence, the correctness of the evaluation procedure follows.
Remark: We observe that in the above construction, the ring elements $i\mathcal{O}$ encodes are chosen according to a distribution depending only on the ring $\mathcal{R}$ and the obfuscated program P; we denoted it as $\mathcal{D}_\lambda(\mathcal{R}, P)$. Moreover, the these elements are encoded under a fixed sequence of levels $\vec{\mathbf{v}}$ independent of $\mathcal{R}$ and P. This fact will be instrumental in the security proof later.

The Ideal Graded Encoding Oracle $\mathcal{O}$: The oracle is fully specified by its ring distribution $\tilde{\mathcal{R}}$ and legality predicate g.

Legality Predicate. We consider a predicate $g_\lambda = g_\lambda[\mathbf{v}_{zt}, V]$ with the zero-testing level $\mathbf{v}_{zt}$ and the set of levels V $(= \{\mathbf{v}^*, \mathbf{v}_s^k, \mathbf{v}^{c+1}, \hat{\mathbf{v}}_{s,o}^k, \hat{\mathbf{v}}_o, \bar{\mathbf{v}}\})$ that the obfuscator $i\mathcal{O}$ uses for encoding elements hardwired in; on input a polynomial and a sequence of levels $(p, \vec{\mathbf{v}})$, it checks i) whether every level in $\vec{\mathbf{v}}$ is in V, and ii) whether p is $(\vec{\mathbf{v}}, \mathbf{v}_{zt})$-respecting and outputs 1 iff both conditions hold.

We now show that this oracle is has degree $O(td(\lambda) + c(\lambda)))$. Below, by total degree of a subset of variables $a_1, \cdots, a_l$ in a polynomial, we mean the degree of a, when replacing all $a_i = a$ in the polynomial.

Claim 5. *$\mathcal{O} = \mathcal{O}_\lambda^{g,\tilde{\mathcal{R}}}$ is a degree-$O(td(\lambda) + c(\lambda))$ ideal graded encoding oracle.*

Proof. Towards this, fix a $\lambda \in \mathbb{N}$, $td = td(\lambda)$, $c = c(\lambda)$ and predicate $g = g_\lambda$ with levels $\mathbf{v}_{zt}$ and V hardwired in. Consider an arbitrary pair $(p, \vec{\mathbf{v}})$. We show that if $g(p, \vec{\mathbf{v}}) = 1$, then the degree of p is bounded by $O(td(\lambda) + c(\lambda))$.

By definition of g, if $g(p, \vec{\mathbf{v}}) = 1$, p must be $(\vec{v}v, \mathbf{v}_{zt})$-respecting, that is, $\mathsf{level}(p, \vec{\mathbf{v}}, \gamma) = \mathbf{v}_{zt}$ (where p is viewed as an arithmetic circuit with output wire γ). By definition of level, every monomial m in p also reaches the same level $\mathbf{v}_{zt}$. We now show that in order to reach level $\mathbf{v}_{zt}$, the degree of different variables in m must satisfy the following conditions:

1. Variables associated with levels $\bar{\mathbf{v}}, \bar{\mathbf{v}} + D\mathbf{v}^*$ has total degree 1, so that the final level would have exactly 1 in position $(1, c+2)$ in the matrix.

2. For every $o \in \Gamma$, variables associated with levels $\{\hat{\mathbf{v}}_o, \hat{\mathbf{v}}_o + \mathbf{v}^*\}_{o \neq o^*}$ has total degree 1, so that the final level would have exactly 1 at position $(q+1, c+1)$. Let o^* be the index such that level $\hat{\mathbf{v}}_{o^*}$ or $\hat{\mathbf{v}}_{o^*} + \mathbf{v}^*$ appears in m.

3. For every $k \in [c]$, variables associated with levels $\{\hat{\mathbf{v}}_{s,o}^k, \hat{\mathbf{v}}_{s,o}^k + \mathbf{v}^*\}_{s, o \neq o^*}$ do not appear. As otherwise, $\hat{\mathbf{v}}_{s,o}^k + \hat{\mathbf{v}}_{o^*} \geq \mathbf{v}_{zt}$ (because of the vector in the level).

4. For every $k \in [c]$, variables associated with levels $\{\hat{\mathbf{v}}_{s,o^*}^k, \hat{\mathbf{v}}_{s,o^*}^k + \mathbf{v}^*\}_s$ have total degree 1 so that the final level would have exactly 1 at position $(q+1, k)$. Let s^k be the index such that level $\hat{\mathbf{v}}_{s^k,o^*}^k$ or $\hat{\mathbf{v}}_{s^k,o^*}^k + \mathbf{v}^*$ appears in m.

5. For every $k \in [c]$, variables associated with levels $\{\mathbf{v}_s^k, \mathbf{v}_s^k + \mathbf{v}^*\}_{s \neq s^k}$ do not appear, as otherwise, $\mathbf{v}_s^k + \hat{\mathbf{v}}_{s^k,o^*}^k + \hat{\mathbf{v}}_{o^*} \geq \mathbf{v}_{zt}$ for $s \neq s^k$.

6. For every $k \in [c]$, variables associated with levels $\mathbf{v}_{s^k}^k, \mathbf{v}_{s^k}^k + \mathbf{v}^*$ has total degree $\mathbf{t}_{o^*}[k]$, so that, when added with $\hat{\mathbf{v}}_{o^*}$ and $\hat{\mathbf{v}}_{s^k,o^*}^k$, the final level will reach value M at position (s^k, k).

7. Variables associated with levels $\mathbf{v}^{c+1}, \mathbf{v}^{c+1} + \mathbf{v}^*$ has total degree $\mathbf{t}_{o^*}[c+1]$, so that, when added with $\hat{\mathbf{v}}_{o^*}$, the final level would reach value M at position $(1, c+1)$

8. Variables associated with level $\mathbf{v}^*$ has total degree $\leq D$, as otherwise $D\mathbf{v}^* \geq \mathbf{v}_{zt}$.

Therefore the total degree of m is bounded by $2 + c + \Sigma_{k \in [c+1]} \mathbf{t}[k] + D \leq 5(td + c)$ as desired. (Recall that $D = d + c + 1$, and by Fact 1, type degree td is no smaller than the degree d.) ☐

Ring Distribution. We use the same ring distribution as [AB15], recalled below.

Definition 20 (Admissible Distributions on Composites and Rings [AB15]). *An ensemble of probability distributions $\{\mathcal{N}_\tau\}$ is τ-admissible if $\mathcal{N}_\tau$ samples a $\mathrm{poly}(\tau)$-bit integer N with the property that the min-entropy of every prime factor of N is at least $\Omega(\tau)$. An ensemble of probability distributions over rings $\{\tilde{\mathcal{R}}_\tau\}$ is τ-admissible if a ring $\mathcal{R}$ sampled from the distribution satisfy $\mathcal{R} \cong \mathbb{Z}_N$ and the random variable N is τ-admissible.*

The reason we use this distribution of rings is the same as [AB15]: Every small fixed integer x is likely to be co-prime to a randomly sampled composite number $y \overset{\$}{\leftarrow} \mathcal{N}_k$ (Lemma 5.6 in [AB15]), and hence for an admissible ring distribution, any fixed (short) list of (small) integers is unlikely to hit *non-invertible*, or *non-unit* ring element. We cite the corollary of Lemma 5.6 in [AB15], which will be important for the analysis of our IO construction.

Corollary 3 ([AB15]). *Let x be an integer such that $|x| \leq 2^{\mathrm{poly}(\lambda)}$. Let $R \cong \mathbb{Z}_N$ be a ring where N is sampled from some $(\log L + \omega(\log \lambda) + t(\lambda))$-admissible distribution. Then, the probability that x is a non-unit in R is $\mathrm{negl}(\lambda)/L2^{t(\lambda)}$.*

Claim 6. *Let $\mathcal{R} \cong \mathbb{Z}_N$ be a ring where N is sampled from some $(\log L + \omega(\log \lambda) + t(\lambda))$-admissible distribution. For every subgroup $\mathbb{Z}_P$ of $\mathcal{R}$, and every multivariate polynomial q of total degree $d = \mathrm{poly}(\lambda)$, it holds that,*

$$\Pr[q(\mathcal{U}_R, \cdots, \mathcal{U}_R) = 0 \textit{ over } \mathbb{Z}_P] \leq \mathrm{negl}(\lambda)/L2^{t(\lambda)}$$

Proof. Consider any subgroup $\mathbb{Z}_P$ of $\mathcal{R}$; $\mathbb{Z}_P \cong \mathbb{Z}_{p_1} \times \cdots \times \mathbb{Z}_{p_\sigma}$ where $p_1, \cdots, p_\sigma$ are the prime factors of P. If a polynomial q evaluates to 0 over $\mathbb{Z}_P$, it must evaluate to 0 over all subgroups of $\mathbb{Z}_P$. Thus,

$$\Pr[q(\mathcal{U}_R, \cdots, \mathcal{U}_R) = 0 \text{ over } \mathbb{Z}_P] \leq \Pr[q(\mathcal{U}_R, \cdots, \mathcal{U}_R) = 0 \text{ over } \mathbb{Z}_{p_1}]$$

Since N is from a $(\log L + \omega(\log \lambda) + t(\lambda))$-admissible distribution, p_1 has min entropy $\Omega(\log L + \omega(\log \lambda) + t(\lambda))$. By Schwartz-Zippel,

$$\Pr[q(\mathcal{U}_R, \cdots, \mathcal{U}_R) = 0 \text{ over } \mathbb{Z}_{p_1}] \leq \frac{d}{p_1} \leq \frac{\mathrm{negl}(\lambda)}{L2^{t(\lambda)}} .$$

□

We set the ring distribution $\tilde{\mathcal{R}}$ of $\mathcal{O}$ to an $\Omega(\log L(\lambda) + \omega(\log \lambda) + t(\lambda))$-admissible distribution, where $L(\lambda) = 2^{\mathrm{poly}(S(\lambda))}$. (Recall that $S(\lambda)$ is the size of the universal arithmetic circuit U of family $\mathcal{P}_\lambda$.) The polynomial t is set according to the security requirement, in particular, when $t(\lambda) = 0$, our IO scheme satisfies polynomial indistinguishability security, whereas when $t(\lambda) = \lambda^\varepsilon$, the IO scheme satisfies sub-exponential security.

6.5 Efficiency of Our IO Scheme

It is easy to see that the number of encodings in the obfuscated program $\hat{P}$ is bounded by $\mathrm{poly}(S(\lambda))$. The size of each encoding (i.e., the random handlers chosen by the oracle) is bounded by $\lambda\ell(\lambda)$, where $\ell(\lambda)$ is an upper bounded on the length of elements in rings sampled from $\tilde{\mathcal{R}}$, which is bounded again by $\mathrm{poly}(\log L + \omega(\log \lambda) + t(\lambda)) = \mathrm{poly}(\lambda, S(\lambda))$. Moreover, the ℓ_1-norm of $\mathbf{v}_{zt}$ (as well as all levels $\leq \mathbf{v}_{zt}$) is also bounded by $\mathrm{poly}(\lambda, S(\lambda))$ for sufficiently large λ. It is easy to check that all poly above are fixed universal polynomials. Therefore, the size of obfuscation is bounded by $p(\lambda, S(\lambda))$ for a universal polynomial, which satisfies the requirement in Theorem 6.

6.6 Unbounded Simulation Security

To show the security of $(i\mathcal{O}, \mathsf{Eval})$, we need to show that for every polynomial-sized adversary $\mathcal{A}$, there exists a randomized unbounded-time simulator $\mathcal{S}$, such that for every $\lambda \in \mathbb{N}$ and $P \in \mathcal{P}_\lambda$, $\mathcal{S}^P(1^\lambda)$ can simulate the view of $\mathcal{A}^{\mathcal{O}}(\hat{P})$ where $\hat{P} \xleftarrow{\$} i\mathcal{O}^{\mathcal{O}}(1^\lambda, P)$, with only oracle access to P, and the distinguishing gap between the real and simulated view is at most $\mu'(\lambda)$. We will consider two cases, μ' is negligible or sub-exponentially small.

Recall that in an honest execution, $i\mathcal{O}^{\mathcal{O}}(1^\lambda, P)$ upon receiving the ring $\mathcal{R} \xleftarrow{\$} \tilde{\mathcal{R}}$ from $\mathcal{O}$, obtains from $\mathcal{O}$ the encoding of a sequence of ring element $\vec{a}$ under a sequence of levels $\vec{\mathbf{v}}$, which constitutes the obfuscated program $\hat{P}$; the sequence of levels $\vec{\mathbf{v}}$ is fixed a priori, while the ring elements $\vec{a}$ are sampled from a distribution $\mathcal{D}_\lambda(\mathcal{R}, P)$ depending only on $\mathcal{R}$ and P.

Our simulator $\mathcal{S}^P(1^\lambda)$ internally runs $\mathcal{A}$ and proceeds as follows:

- *Simulating $\hat{P}$:* $\mathcal{S}$ does not know P, hence it cannot sample from $\mathcal{D}_\lambda(\mathcal{R}, P)$. But, since encodings from $\mathcal{O}$ are simply random handles (of appropriate length), $\mathcal{S}$ emulates them by sampling a sequence of random handles $\vec{h}$ on its own and records for each handle h_i a tuple $(\star, \mathbf{v}_i, h_i)$, where $\mathbf{v}_i$ is the i^{th} level in $\vec{\mathbf{v}}$. It then feeds $\tilde{P} = \vec{h}$ to $\mathcal{A}$.

- *Simulating the Zero-Testing Procedure of $\mathcal{O}$:* Upon receiving from $\mathcal{A}$ an oracle-query q which is a polynomial, $\mathcal{S}$ simulates the answer of $\mathsf{OEval}(q)$ as follows.
  - If q is not a polynomial over variables $\vec{h}$, return 0. [4]
  - If the legality test fails, that is, $g_\lambda[\mathbf{v}_{zt}, V](q, \vec{\mathbf{v}}) = 0$, return 0. Otherwise, continue to next step; since every level in $\vec{\mathbf{v}}$ is in V by definition, q passes the legality test iff it is $(\vec{\mathbf{v}}, \mathbf{v}_{zt})$-respecting.
  - $\mathcal{S}$ invokes another deterministic simulation procedure $b \xleftarrow{\$} \mathsf{Sim}^P(1^\lambda, q)$, called the *algebraic simulator* (described shortly below), which simulates the output of q evaluated on elements $\vec{a} \xleftarrow{\$} D_\lambda(\mathcal{R}, P)$ using only oracle access to $P(\star)$. $\mathcal{S}$ then feeds the output b of Sim to $\mathcal{A}$.

It is easy to see that $\mathcal{S}$ emulates the view of $\mathcal{A}$ perfectly up to the last step, and the indistinguishability of the simulated view by $\mathcal{S}$ depends on the "correctness" of the algebraic simulator Sim, namely, let $\mu = \mathrm{negl}(\lambda)/2^{t(\lambda)}$ for some polynomial t.

$\mu(\lambda)$**-Algebraic security** [**AB15**]**:** For every ensemble of polynomial-sized polynomials $\{q\}_\lambda$ where q is $(\vec{\mathbf{v}}, \mathbf{v}_{zt})$-respecting,

$$\left|\Pr[\vec{a} \xleftarrow{\$} \mathcal{D}_\lambda(\mathcal{R}, P) \ : \ P_{zero}(q(\vec{a})) = 1] - \Pr[\mathsf{Sim}^P(1^\lambda, q) = 1]\right| \leq \mu(\lambda) \ . \tag{8}$$

where P_{zero} is a predicate that tests whether the input is zero.

The algebraic security shows that for a single query q, Sim emulates whether the output of q is zero or not correctly with overwhelming probability (over the randomness for choosing the encoded elements). Then by a union bound, for all (polynomial number of) queries from $\mathcal{A}$, $\mathcal{S}$ emulates the answers from $\mathcal{O}$ correctly with $\mu'(\lambda) = \mathrm{poly}(\lambda)\mu(\lambda)$ probability, and hence the simulated view is $\mu'(\lambda)$-statistically close to the real view.

Next, we turn to constructing the algebraic simulator Sim. In fact, we will construct a simulation procedure Sim that satisfies a even stronger correctness property, that is, Sim not only emulates

[4]For convenience, if a variable does not appear in q, we still consider it an input variable, just with degree 0.

outputs of polynomials that reach the zero-testing level $\mathbf{v}_{zt}$ (as the legality test requires), but also these that reach any level $\mathbf{v}' \leq \mathbf{v}_{zt}$. Formally, we prove the following:

Lemma 7. *There is a deterministic unbounded algebraic simulator* Sim *satisfying*

$\mu(\lambda)$**-Strong algebraic security:** *For every ensemble of polynomial-sized polynomials* $\{q\}_\lambda$ *satisfying that* q *is* $(\vec{\mathbf{v}}, \mathbf{v}')$*-respecting for some* $\mathbf{v}' \leq \mathbf{v}_{zt}$*,* Sim *satisfies equation* (8)*.*

Remark: We remark that the benefit of proving the strong algebraic security is twofold: 1) By [AB15], canonical IO schemes satisfying strong algebraic security is secure even if the ideal graded encoding model gives unique encoding for every ring element (See Remark 2 and Lemma 3.7 in [AB15]); 2) From the analysis below showing strong algebraic security it is easy to observe that our IO scheme (inherited from [AB15]) satisfies that generic attackers can only produce encodings of zero at the zero testing level; see Lemma 9. Though this property does not completely thwart zeroizing attacks on graded encoding schemes; it does make it more difficult as discussed in [CGH+15] and seems to be the best achievable if the evaluation of IO scheme relies on zero-testing.

Construction of Sim**:** Fix λ, $P \in \mathcal{P}_\lambda$, $\tilde{\mathcal{R}} = \tilde{\mathcal{R}}_\lambda$, $\mathcal{D} = \mathcal{D}_\lambda$, and a poly$(\lambda)$-sized polynomial q that is $(\vec{\mathbf{v}}, \mathbf{v}_{zt})$-respecting.

On input 1^λ and q, Sim needs to emulate the output of q evaluated over variables $\vec{a}$ sampled from the distribution $\vec{a} \xleftarrow{\$} \mathcal{D}(\mathcal{R}, P)$, $\mathcal{R} \xleftarrow{\$} \tilde{\mathcal{R}}$. Recall that by the construction of $i\mathcal{O}^{\mathcal{O}}(\lambda, P)$; these variables $\vec{a}$ are in turn described by variables $\vec{r}$ and $\vec{w}$. Thus, it is equivalent to think of q as a polynomial over $\vec{r}$ and $\vec{w}$, and $\mathcal{D}(\mathcal{R}, P)$ a distribution sampling $\vec{r}, \vec{w}$. The algebraic simulator $\mathsf{Sim}^P(1^\lambda, q)$ proceeds as follows:

1. *Decomposition:* Sim "decomposes" P as a sum of terms of the form $M(\vec{r}) \cdot Q(\vec{w})$ where M is a monomial, and Q is a polynomial, namely, $P = \Sigma_i M_i(\vec{r}) \cdot Q_i(\vec{w})$. There are at most $L = 2^{\mathrm{poly}(S(\lambda))}$ terms in the summation, since

   Claim 7. *There are at most* $L = 2^{\mathrm{poly}(S(\lambda))}$ *distinct* $M(\vec{r})$ *monomials.*

   Proof. Since q is $(\vec{\mathbf{v}}, \mathbf{v}')$-respecting for $\mathbf{v}' \leq \mathbf{v}_{zt}$, it is easy to see that the degree of q is bounded by $|\mathbf{v}_{zt}|_1$, and so is the degree of any monomial $M(\vec{r})$ in q. Therefore, the number of distinct monomials M is bounded by $L = |\vec{r}|^{|\mathbf{v}_{zt}|_1}$. By construction of $i\mathcal{O}$, the number of r variables, as well as $|\mathbf{v}_{zt}|_1$, is bounded by poly$(S(\lambda))$; hence, we have that $L = 2^{\mathrm{poly}(S(\lambda))}$. $\square$

2. *Zero-Testing each Q polynomial:* For each term $M(\vec{r}) \cdot Q(\vec{w})$, Sim emulates whether the output of $Q(\vec{w})$ is zero or non-zero. First, we show that Q has the following structure.

   Lemma 8. *There is an* $o^* \in \Gamma$*, a constant* a*, and a polynomial* $Q'(\vec{w})$*, such that,*

   $$Q(\vec{w}) = a \cdot \bar{w}_{o^*} - Q'(\vec{w}) \ .$$

   Moreover, Q' *is free of variables* $\{\bar{w}_o\}_{o \in \Gamma}$*, and the degree of each of variables* $\hat{w}$ *and* $\{\hat{w}^k\}_k$ *is at most 1*

   Given the above structure, Sim considers three cases:

   - **Case 1:** $\left(\hat{w} \prod_{k \in [c]} \hat{w}^k\right) \nmid Q'(\vec{w})$.

If Case 1 do not hold, that is, $(\hat{w}\prod_{k\in[c]}\hat{w}^k)|Q'(\vec{w})$. By Lemma 8, there is another polynomial $Q''(\vec{w})$ that is free of all $\{\bar{w}_o\}_o$, $\hat{w}$, and $\{\hat{w}^k\}_k$ variables, such that,

$$Q(\vec{w}) = a\bar{w}_{o^*} - \left(\hat{w}\prod_{k\in[c]}\hat{w}^k\right)\cdot Q''(\vec{w})$$
$$= \left(\hat{w}\cdot\prod_{k\in[c]}\hat{w}^k\right)\cdot(a\,(\bar{y}_{o^*},\ 1,\ 0,\ \cdots,0)) - \left(\hat{w}\prod_{k\in[c]}\hat{w}^k\right)\cdot Q''(\vec{w}) \qquad (\ddagger)$$

- **Case 2:** Q has the structure of $(\ddagger)$, but for every possible input $x\in\Sigma^c$,

$$Q''(\vec{w}) \neq a\cdot U_{o^*}(\{w^1_{x^1,j}\},\cdots,\{w^{c+1}_{x^{c+1},j}\},\{w^{c+1}_j\})$$

- **Case 3:** Q has the structure of $(\ddagger)$, and there is an input $x\in\Sigma^c$,

$$Q''(\vec{w}) = a\cdot U_{o^*}(\{w^1_{x^1,j}\},\cdots,\{w^{c+1}_{x^{c+1},j}\},\{w^{c+1}_j\})$$

In Case 1 and 2, Sim determines that the output of Q is *non-zero*. We show in Claim 8 and 10 below that this emulation is correct with high probability $1-\mathsf{err}(\lambda)$ where $\mathsf{err} = \mathrm{negl}(\lambda)/L(\lambda)2^{t(\lambda)}$.

In Case 3, Sim queries its oracle $P(\star)$ on input x, obtaining output y; it determines that the output of Q is *zero* iff $y_o = 0$. We show in Claim 11 below that this emulation is correct with high probability $1-\mathsf{err}(\lambda)$.

3. *Summarizing:* If for *every* term $M(\vec{r})Q(\vec{w})$ in q, the output of $Q(\vec{w})$ is determined to be zero in the previous step, Sim returns 1 (meaning that the output of q is zero). Otherwise, if for *any* term $M(\vec{r})Q(\vec{w})$ in q, the output of $Q(\vec{w})$ is determined to be non-zero, Sim returns 0 (meaning that the output of q is non-zero).

Strong Algebraic Security of Sim: By Claim 8 to 11, for each term $M(\vec{r})Q(\vec{w})$, Sim determines correctly whether Q evaluates to zero or non-zero with probability $1-\mathsf{err}(\lambda)$. By a union bound, Sim determines correctly for all Q whether they are zero or non-zero, with probability $1-L\times\mathsf{err}(\lambda)$. Conditioned on this happening, if all Q evaluates to zero, then Sim correctly replies that the output of q is zero. In the other case, q can be viewed as a polynomial over $\vec{r}$ with coefficients $Q(\vec{w})$'s, and if some polynomial Q evaluates to non-zero, this polynomial over $\vec{r}$ is not identically zero. Then by Claim 6, when the values of $\vec{r}$ are randomly chosen, the probability that q evaluates to zero is bounded by $\mathsf{err}(\lambda)$. Overall, by union bound, the error probability of Sim is $\mathrm{negl}(\lambda)/2^{t(\lambda)}$.

Next, we prove Lemma 8 and the Claims.

Proof of Lemma 8. Towards the lemma, we prove the following sequence of structural facts of Q, one building upon another.

- The total degree of variables $\{\bar{w}_o\}_{o\in\Gamma}$ is ≤ 1.

  Suppose not and for some $o\neq o'$, both $\bar{w}_o$ and $\bar{w}_{o'}$ appear in Q. By construction of $i\mathcal{O}$, every w variable appear in El-Gamal form, and hence, both $\bar{r}_o$ and $\bar{r}_{o'}$ are in $M(\vec{r})$. But, they all correspond to level $\bar{\mathbf{v}}$, and $2\bar{\mathbf{v}}\geq\mathbf{v}_{zt}$ (because of column $c+1$ in the matrix of the level).

  Let o^* be the index such that $\bar{w}_{o^*}$ appears in Q.

- It holds that $Q(\vec{w}) = a \cdot \bar{w}_{o^*} - Q'(\vec{w})$ and Q' is free of variables $\{\bar{w}_o\}$.

  Suppose not. Then in $Q'(\vec{w})$, $\bar{w}_{o^*}$ appears and is multiplied with another w variable. However, since the level of $\bar{w}_{o^*}$ satisfies $\bar{\mathbf{v}} \geq D\mathbf{v}^*$, and the level of any w variable is $\geq \mathbf{v}^*$, they cannot be multiplied together, or else the level would exceed $\mathbf{v}_{zt}$ (because of position $(q+1, c+2)$ in the matrix of the level).

- The degree of $\hat{w}$ is at most 1 in Q'.

  Suppose not and the quadratic term $\hat{w}^2$ appears in Q'. Recall that $\hat{w}$ appears under levels $\{\hat{\mathbf{v}}_o\}_{o \in \Gamma}$. However, for any o and o', $\hat{\mathbf{v}}_o + \hat{\mathbf{v}}_{o'} \geq \mathbf{v}_{zt}$ (because of position $(q+1, c+1)$ in the matrix of the level).

- For every $k \in [c]$, the degree of variables $\hat{w}^k$ is at most 1 in Q'.

  Suppose not and the quadratic term $(\hat{w}^k)^2$ appears in Q'. Recall that $\hat{w}^k$ appears under levels $\{\hat{\mathbf{v}}^k_{s,o}\}_{s \in \Sigma, o \in \Gamma}$. However, for any s, o and s', o', $\hat{\mathbf{v}}^k_{s,o} + \hat{\mathbf{v}}^k_{s',o'} \geq \mathbf{v}_{zt}$ (because of position $(q+1, k)$ in the matrix of the level).

Summing up the above structural observations, we conclude the lemma. $\square$

Claim 8. *The following holds in Case 1,*

$$\Pr[\mathcal{R} \xleftarrow{\$} \tilde{\mathcal{R}}, (\vec{r}, \vec{w}) \xleftarrow{\$} \mathcal{D}(\mathcal{R}, P) \ : Q(\vec{w}) = 0] = \frac{\mathrm{negl}(\lambda)}{L(\lambda)2^{t(\lambda)}}$$

Proof. Case 1 states that $(\hat{w} \prod_{k \in [c]} \hat{w}^k) \nmid Q'(\vec{w})$. Below we prove that if there is a $k \in [c]$ such that $\hat{w}^k$ does not divide Q', then Q outputs zero with very small probability. The same proof also applies to the case when $\hat{w}$ does not divide Q'.

Recall that $Q(\vec{w}) = a\bar{w}_{o^*} - Q'(\vec{w})$. Consider the evaluation of Q over the $k+2$th sub-ring, $Q(\vec{w})[\![k+2]\!]$. Since $\bar{w}_{o^*}[\![k+2]\!] = 0$ for all $k \in [c+1]$, we have that $Q(\vec{w})[\![k+2]\!] = Q'(\vec{w})[\![k+2]\!]$. Since $Q'(\vec{w})[\![k+2]\!] = Q'[\![k+2]\!](\vec{w}[\![k+2]\!])$ (i.e., the evaluation of $Q'[\![k+2]\!]$ over the $\vec{w}[\![k+2]\!]$). Recall that all w variables, except $\hat{w}^k$, contain random ρ element ($= \{\rho^{c+1}_{j,k}\}$, $\{\rho^{k'}_{s,j,k}\}$, $\{\hat{\rho}^{k'}_k\}_{k \neq k'}$, $\hat{\rho}_k$) in the $k+2$th sub-ring.

Since Q' is not identically zero (or else Case 1 does not hold), it must contain one non-zero coefficient α. By Corollary 3, except with probability $\mathrm{negl}(\lambda)/L2^{t(\lambda)}$ over the sampling of the ring $\mathcal{R} \xleftarrow{\$} \tilde{\mathcal{R}}$, α is a unit and $\alpha[\![k+2]\!]$ is non-zero. Therefore $Q'[\![k+2]\!]$ is also not identically zero.

Therefore, by Claim 6, the probability that $Q'[\![k+2]\!]$ evaluates to zero over randomly chosen ρ variables in the $k+2$th sub-ring $\mathcal{R}_{k+2}$ is $\mathrm{negl}(\lambda)/L2^{t(\lambda)}$. $\square$

Next, before analyzing Case 2 and 3, we first prove that when Case 1 does not hold, that is, $(\hat{w} \prod_{k \in [c]} \hat{w}^k) | Q'(\vec{w})$, Q' must be consistent with some input $x = x^1, \cdots, x^c$, meaning that Q' is free of other variables $\{w^k_{s,j}\}_{s \neq x^k, j \in [\ell]}$ corresponding to other input values.

Claim 9. *If $(\hat{w} \prod_{k \in [c]} \hat{w}^k) | Q'(\vec{w})$, there exists some input $x = x^1, \cdots, x^c$, such that, Q' is free of variables $\{w^k_{s,j}\}_{s \neq x^k, j \in [\ell]}$.*

Proof. Suppose not. Then Q' is non-zero and there is a $k \in [c]$, $s \neq s'$, and j, j' such that both $w^k_{s,j}$ and $w^k_{s',j'}$ appear in Q'. Recall that these two variables appear under levels $\mathbf{v}^k_{s,j}, \mathbf{v}^k_{s',j'}$.

Since the variables $\hat{w}$ and $\hat{w}^k$ divide Q' and appear under levels $\{\hat{\mathbf{v}}_o\}$ and $\{\hat{\mathbf{v}}^k_{s,o}\}$ respectively, there exist some indexes s'', o'' and $\bar{o}$ such that the level of Q' is at least $\geq \hat{\mathbf{v}}_{s'',o''} + \hat{\mathbf{v}}_{\bar{o}}$. Since

$\hat{\mathbf{v}}_{s'',o''} + \hat{\mathbf{v}}_{\bar{o}} \geq \mathbf{v}_{zt}$ for any $o'' \neq \bar{o}$ (because of the vector in the levels), we conclude that $\bar{o} = o''$. But, this leads to a contradiction: When adding additionally the levels $\mathbf{v}^k_{s,j}, \mathbf{v}^k_{s',j'}$, $s \neq s'$, the final level exceeds $\mathbf{v}_{zt}$, $\hat{\mathbf{v}}_{s'',o''} + \hat{\mathbf{v}}_{o''} + \mathbf{v}^k_{s,j}, \mathbf{v}^k_{s',j'} \geq \mathbf{v}_{zt}$. □

Claim 10. *The following holds in Case 2,*

$$\Pr[\mathcal{R} \xleftarrow{\$} \tilde{\mathcal{R}}, (\vec{r}, \vec{w}) \xleftarrow{\$} \mathcal{D}(\mathcal{R}, P) \ : Q(\vec{w}) = 0] = \frac{\text{negl}(\lambda)}{L(\lambda)2^{t(\lambda)}}$$

Proof. In Case 2, Q has the structure of (‡),

$$Q(\vec{w}) = \left(\hat{w} \cdot \prod_{k \in [c]} \hat{w}^k\right) \cdot (a\,(\bar{y}_{o^*},\ 1,\ 0,\ \cdots, 0)) - \left(\hat{w} \prod_{k \in [c]} \hat{w}^k\right) \cdot Q''(\vec{w})$$

By Claim 9, Q'' must be consistent with some input x, but,

$$Q''(\vec{w}) \neq a \cdot U_{o^*}(\{w^1_{x^1,j}\}, \cdots, \{w^{c+1}_{x^{c+1},j}\}, \{w^{c+1}_j\}) \ . \qquad (*)$$

Consider the evaluation of Q in the first sub-ring:

$$\begin{aligned} Q(\vec{w})[\![1]\!] &= \left(\hat{w} \cdot \prod_{k \in [c]} \hat{w}^k\right)[\![1]\!] \cdot \left(a \cdot \bar{y}_{o^*} - Q''(\vec{w})[\![1]\!]\right) \\ &= \left(\hat{w} \cdot \prod_{k \in [c]} \hat{w}^k\right)[\![1]\!] \cdot \Big(a[\![1]\!] \cdot U_{o^*}(\{y^1_{x^1,j}\}, \cdots, \{y^{c+1}_{x^{c+1},j}\}, \{y^{c+1}_j\}) - Q''(\vec{w})[\![1]\!]\Big) \end{aligned}$$

By equation $(*)$, $Q''(\cdots) - aU_{o^*}(\cdots)$ is not identically zero and has a non-zero coefficient α (where $\cdots$ represents the same variables in $(*)$ and is omitted for convenience). By Corollary 3, except with probability $\text{negl}(\lambda)/L2^{t(\lambda)}$, α is a unit, and $\alpha[\![1]\!]$ is non-zero. Therefore, the polynomial in the first sub-ring, $(Q''(\cdots) - aU_{o^*}(\cdots))[\![1]\!]$, is not identically zero. Since all variables $\{w^k_{s,j}\}$, $\{w^{c+1}_j\}$ contain random elements $\{y^k_{s,j}\}$, $\{y^{c+1}_j\}$ in the first sub-ring (and Q'' is free of any $\bar{w}, \hat{w}$ variables), by Claim 6, the probability that $(Q''(\cdots) - aU_{o^*}(\cdots))[\![1]\!]$ evaluates to zero over randomly chosen $\{y^k_{s,j}\}$, $\{y^{c+1}_j\}$ in the first sub-ring $\mathcal{R}_1$ is $\text{negl}(\lambda)/L2^{t(\lambda)}$. Hence, the probability that Q evaluates to zero is $\leq \text{negl}(\lambda)/L2^{t(\lambda)}$. □

Claim 11. *The following holds in Case 3:*

$$\begin{aligned} &\textit{If } U_o(x, P) = 1\ ,\ \Pr[\mathcal{R} \xleftarrow{\$} \tilde{\mathcal{R}}, (\vec{r}, \vec{w}) \xleftarrow{\$} \mathcal{D}(\mathcal{R}, P) \ : Q(\vec{w}) = 0] = \frac{\text{negl}(\lambda)}{L(\lambda)2^{t(\lambda)}} \\ &\textit{If } U_o(x, P) = 0\ ,\ \Pr[\mathcal{R} \xleftarrow{\$} \tilde{\mathcal{R}}, (\vec{r}, \vec{w}) \xleftarrow{\$} \mathcal{D}(\mathcal{R}, P) \ : Q(\vec{w}) = 0] = 1 \end{aligned}$$

Proof. In Case 3, Q has the structure of (‡), and there is an input $x \in \Sigma^c$,

$$Q''(\vec{w}) = a \cdot U_{o^*}(\{w^1_{x^1,j}\}, \cdots, \{w^{c+1}_{x^{c+1},j}\}, \{w^{c+1}_j\})$$

First notice that a must be non-zero, or else Q is identically zero. Then by Claim 3, a is a unit except with probability $\text{negl}(\lambda)/L2^{t(\lambda)}$.

By definition $Q(\vec{w})$ evaluates to zero on all sub-rings except the second. Therefore, it suffices to test whether the $Q(\vec{w})[\![2]\!]$ is zero or non-zero. On the second sub-ring,

$$Q(\vec{w})[\![2]\!] = a[\![2]\!] \cdot \left(\hat{w} \cdot \prod_{k\in[c]} \hat{w}^k\right)[\![2]\!] \cdot \left(1 - U_{o^*}(x^1, \cdots, x^{c+1}, P)\right)$$

Therefore, if $U_{o^*}(x, P) = 1$, $Q(\vec{w})[\![2]\!]$ equals to zero with probability 1 and so does $Q(\vec{w})$. Otherwise, in the case $U_{o^*}(x, P) = 0$, $Q(\vec{w})[\![2]\!]$ is a non-zero polynomial (with a non-zero coefficient $a[\![2]\!]$) over random $\hat{w}[\![2]\!]$ and $\{\hat{w}^k[\![2]\!]\}$, which by Claim 6 is non-zero except with probability $\mathrm{negl}(\lambda)/L2^{t(\lambda)}$, and so is $Q(\vec{w})$. □

No Low Level Encodings of Zeros We show that it follows from the above analysis that any generic attacker given an obfuscated program output by our IO scheme cannot generate encodings of zero at any level below $\mathbf{v}_{zt}$.

Lemma 9. *For every ensemble of $\{P\}_\lambda$ where $P \in \mathcal{P}_\lambda$, and every PPT oracle adversary $\mathcal{A}$, the probability that*

$$Pr[\hat{P} \xleftarrow{\$} i\mathcal{O}^{\mathcal{O}}(1^\lambda, P),\ q \xleftarrow{\$} \mathcal{A}^{\mathcal{O}}(\hat{P})\ :\ q \textit{ is } (\mathbf{v}, \mathbf{v}')\textit{-respecting for } \mathbf{v}' \le \mathbf{v}_{zt} \textit{ and } P_{zero}(q(\hat{P})) = 1] \le \mathrm{negl}(\lambda)/2^{t(\lambda)}$$

Proof. The proof of Lemma 7 above shows that for a polynomial q, if any of the terms $M(\vec{r})Q(\vec{w})$ in q belongs to Case 1 or 2, the probability that q evaluates to zero is bounded by $\mathrm{negl}(\lambda)/2^{t(\lambda)}$. The only case where q evaluates to zero with noticeable probability is if all its terms $M(\vec{r})Q(\vec{w})$ fall into Case 3. However, we show that in this case, q is $(\vec{\mathbf{v}}, \vec{\mathbf{v}_{zt}})$-respecting, which implies that a generic attacker cannot produce any encodings of zero at levels below $\mathbf{v}_{zt}$ except with probability $\mathrm{negl}(\lambda)/2^{t(\lambda)}$.

We show every term $M(\vec{r})Q(\vec{w})$ in Case 3 reach level $\mathbf{v}_{zt}$. This is because

$$Q(\vec{w}) = a \cdot \hat{w}(\prod \hat{w}^k)(\bar{w}_{o^*} - U_{o^*}(\{w^1_{x^1,j}\}, \cdots, \{w^{c+1}_{x^{c+1},j}\}, \{w^{c+1}_j\}))$$

Therefore, for every w variable in Q with degree d, $M(\vec{r})$ contains the corresponding r variable with the same degree. Hence, the level of $M(\vec{r})Q(\vec{w})$ is at least $\ge \mathbf{v}_{zt} - D\mathbf{v}^*$. On the other hand, since $\bar{w}_{o^*}$ appears, the level of $M(\vec{r})Q(\vec{w})$ is also at least $\ge D\mathbf{v}^*$. These two constraints shows that q reaches $\mathbf{v}_{zt}$. □

7 IO from Constant-Degree Semantically Secure Graded Encodings

We formalize the notion of *distributional semantic security* for composite order GE schemes where the rings are sampled from a distribution; our definition is adapted directly from the notion of semantic security of GE schemes proposed by Pass, Seth and Telang [PST14a], which however focuses on fixed prime order rings. We then show that our IO scheme in ideal GE model in Section 6 can be instantiated in the plain model using distributional semantically secure graded encoding schemes.

7.1 Distributional Semantic Security of GE schemes

Roughly speaking, the semantic security of [PST14a] states the following: Consider two sequences of levels $(\vec{\mathbf{v}}, \vec{\mathbf{w}})$, a zero testing level $\mathbf{v}_{zt}$, and a distribution $\mathbb{E}$ over elements, i.e., elements in a ring $\mathcal{R}$, $(\vec{m}_0, \vec{m}_1, \vec{z})$ satisfying a nice "zero-knowledge" property to generic attackers — for every $((\vec{\mathbf{v}}, \vec{\mathbf{w}}), \mathbf{v}_{zt})$-respecting arithmetic circuits C, the output of C on $(\vec{m}_b, \vec{z})$ for $b \in \{0, 1\}$ is constant with extremely high probability over the choice of elements; then, the encoding of $(\vec{m}_b, \vec{z})$ under levels $(\vec{\mathbf{v}}, \vec{\mathbf{w}})$ is indistinguishable when $b = 0$ or 1 to any computationally efficient attackers.

Our distributional-semantic security require the same security property as the semantic security in [PST14a], but with three difference: 1) The ring $\mathcal{R}$ has composite order rather than prime order. 2) $\mathcal{R}$ is sampled from an admissible distribution $\tilde{\mathcal{R}}$ and the distribution of elements $\mathbb{E}$ only needs to satisfy the "zero-knowledge" property over a randomly chosen ring from the distribution, as opposed to over every ring in the support of the distribution. The second difference is needed since our IO construction in Section 6 crucially relies on the fact that the ring $\mathcal{R}$ is hidden from the adversary, whereas the semantic security of [PST14a] allows the adversary to depend on $\mathcal{R}$. Finally, 3) we allow a bigger error probability (still sub-exponentially small) in the zero-knowledge property than [PST14a].

We remark that though stronger than the plain semantic security, our distributional-semantic security is weaker than another assumption, namely the statistical uber security assumption, proposed in [PST14a]. In particular, statistical uber assumption also considers security for randomly chosen rings, but puts a much weaker constraints on the distribution $\mathbb{E}$ of elements—instead of requiring the zero-knowledge property on $\mathbb{E}$, it only require that (m_b, z) for $b = 0$ or 1 to be indistinguishable to (computationally unbounded) generic attackers who makes a polynomial number of zero-testing queries.

We now formalize distributional-semantic security.

Definition 21 (Valid Level-Respecting Element Sampler)**.** *Let $\tilde{\mathcal{R}}$ be a distribution over rings of size $2^{\ell(\lambda)}$ and let $\{(\vec{\mathbf{v}}, \vec{\mathbf{w}}, \mathbf{v}_{zt})\}_{\lambda \in \mathbb{N}}$ be an ensemble of levels where every $\vec{\mathbf{v}}[i], \vec{\mathbf{w}}[j] \leq \mathbf{v}_{zt}$. We say that a non-uniform PPT machine $\mathbb{E}$ is a $\{(\vec{\mathbf{v}}, \vec{\mathbf{w}}, \mathbf{v}_{zt})\}$-respecting element sampler w.r.t. $\tilde{\mathcal{R}}$ if*

Syntax: *$\mathbb{E}$ on input 1^λ and a ring $\mathcal{R} \in \tilde{\mathcal{R}}(1^\lambda)$ (i.e., in the support of $\tilde{\mathcal{R}}(1^\lambda)$) outputs a pair of sequence of elements $(\vec{m}_0, \vec{m}_1)$ each of length $|\vec{\mathbf{v}}|$ and a sequence of elements $\vec{z}$ of length $|\vec{\mathbf{w}}|$.*

Validity: *There exists a constant ε, such that, for every ensemble of $(\vec{\mathbf{v}}, \vec{\mathbf{w}}, \mathbf{v}_{zt})$-respecting arithmetic circuits $\{C_\lambda\}_{\lambda \in \mathbb{N}}$, it holds that for every $\lambda \in \mathbb{N}$, $C = C_\lambda$, there is a constant c, such that, for every $b \in \{0, 1\}$,*

$$\Pr[\mathcal{R} \xleftarrow{\$} \tilde{\mathcal{R}}(1^\lambda),\ (\vec{m}_0, \vec{m}_1, \vec{z}) \xleftarrow{\$} \mathbb{E}(1^\lambda, \mathcal{R})\ :\ P_{zero}(C(\vec{m}_b, \vec{z})) = c] \geq 1 - \frac{1}{2^{\ell(\lambda)^\varepsilon}} ,$$

where $C(\vec{m}_b, \vec{z})$ is computed over ring $\mathcal{R}$.

Given the notion of valid level-respecting element samplers, the rest of the definition is identical to semantic security in [PST14a].

Definition 22 (μ-Semantic Security)**.** *Let q, c be polynomials. We say that a graded encoding scheme GES supporting ring distribution $\tilde{\mathcal{R}}$ is (c, q)-semantically secure if for every constant $\sigma \in \mathbb{N}$, every polynomial k, every ensemble $\{(\vec{\mathbf{v}}, \vec{\mathbf{w}}, \mathbf{v}_{zt})\}_{\lambda \in \mathbb{N}}$ of levels, where $\vec{\mathbf{v}}$ and $\vec{\mathbf{w}}$ have length $c(\lambda)$ and $q(\lambda)$ respectively, and every $\vec{\mathbf{v}}[i], \vec{\mathbf{w}}[j] \leq \mathbf{v}_{zt}$, every $\{(\vec{\mathbf{v}}, \vec{\mathbf{w}}, \mathbf{v}_{zt})\}$-respecting element sampler $\mathbb{E}$ w.r.t. $\tilde{\mathcal{R}}$, and every non-uniform PPT adversary $\mathcal{A}$, the following holds for every $\lambda \in \mathbb{N}$,*

$$|\Pr[\mathsf{Output}_0(1^\lambda)] - \Pr[\mathsf{Output}_1(1^\lambda)]| \leq \mu(\lambda) ,$$

where $\mathsf{Output}_b(1^\lambda)$ *is* $\mathcal{A}$*'s output in the following game:*

- $(\mathsf{pp}, \mathsf{sp}) \overset{\$}{\leftarrow} \mathsf{InstGen}(1^\lambda, 1^\sigma, 1^k, 1^{\mathbf{v}_{zt}})$*;* sp *describes a* $(\mathcal{R}, \mathbf{v}_{zt})$*-graded encoding scheme.*
- $(\vec{m}_0, \vec{m}_1, \vec{z}) \overset{\$}{\leftarrow} \mathbb{E}(1^\lambda, \mathcal{R})$*.*
- $\vec{A}_b \overset{\$}{\leftarrow} \{\mathsf{Encode}(\mathsf{sp}, \vec{m}_b[i], \vec{\mathbf{v}}[i])\}_{i\in[c(\lambda)]}, \{\mathsf{Encode}(\mathsf{sp}, \vec{z}[j], \vec{\mathbf{w}}[j])\}_{j\in[q(\lambda)]}$*.*
- *Output* $\mathcal{A}(1^\lambda, \mathsf{pp}, \vec{A}_b)$*.*

Below, for convenience, we do not explicitly specify the parameters (c, q), which is clear in the context. We say a graded encoding scheme is semantically secure, if it is μ-semantically secure for every μ that is an inverse polynomial, and is sub-exponentially semantically secure if it is μ-semantically secure for a sub-exponentially small μ.

7.2 Instantiating Our IO Scheme in the Plain Model

We now instantiate our IO scheme in Section 6 using distributional-semantically secure graded encoding schemes.

Theorem 7. *Let* $\{\mathcal{P}_\lambda\}$ *be a circuit class with universal arithmetic circuits* $\{U_\lambda\}$ *of* $c(\lambda)$ *input-types, type degree* $\mathsf{tdeg}(\lambda)$*, and size* $S(\lambda)$*, where* c*,* tdeg*,* S *are polynomials. Let* $\tilde{\mathcal{R}}$ *be a* $\Omega(\log L(\lambda) + \omega(\log\lambda) + t(\lambda))$*-admissible ring distribution, where* $L(\lambda) = 2^{\mathrm{poly}(S(\lambda))}$ *and* $t(\lambda) >> \log L(\lambda) + \omega(\log\lambda)$*.*

Assume the existence of a (sub-exponentially secure) distributional-semantically secure graded encoding scheme GES *supporting* $\tilde{\mathcal{R}}$ *and degree* $d(\lambda) = O(\mathsf{tdeg}(\lambda) + c(\lambda))$*. There is a (sub-exponentially secure) IO for* $\{\mathcal{P}_\lambda\}$ *in the plain model with run time* $u(1^\lambda, S(\lambda))$ *for a universal polynomial* u*.*

Proof. Given such a $\mathsf{GES} = (\mathsf{InstGen}, \mathsf{Encode}, \mathsf{Add}, \mathsf{Sub}, \mathsf{Mult}, \mathsf{isZero})$. The construction of IO for $\{\mathcal{P}_\lambda\}$ in the ideal model in Section 6.4 can be easily modified to use GES:

- The obfuscator $i\mathcal{O}(1^\lambda, P)$ calls the setup algorithm $\mathsf{InstGen}(1^\lambda, 1^{c+3}, 1^{d(\lambda)}, 1^{\mathbf{v}_{zt}})$ to obtain $(\mathsf{sp}, \mathsf{pp})$ (note the level of multilinearity is set to $d(\lambda)$), where sp specifies a $(\mathcal{R}, \mathbf{v}_{zt})$-graded encoding scheme and the $\mathcal{R}$ follows distribution $\tilde{\mathcal{R}}$. The obfuscator then samples from the element distribution $\vec{a} \overset{\$}{\leftarrow} \mathcal{D}_\lambda(\mathcal{R}, P)$ as before, and then encodes the sequence of elements $\vec{a}$ under the a-priori fixed sequence of levels $\vec{\mathbf{v}}$, $\vec{\alpha} \overset{\$}{\leftarrow} \{\mathsf{Encode}(\mathsf{sp}, \vec{a}[i], \vec{\mathbf{v}}[i])\}$. It outputs obfuscated program $\hat{P} = (\mathsf{pp}, \vec{\alpha})$.
- The Evaluator $\mathsf{Eval}(\mathsf{pp}, \hat{P})$, on the other hand, performs the same evaluation over the encodings in $\hat{P}$ as before, but using pp and algorithms Add, Sub, and Mult, and obtain a final encoding $\tilde{Z}_o$ for every output wire o. It then sets $y_o = \mathsf{isZero}(\mathsf{pp}, \tilde{Z}_o)$ as the value of that output wire.

The correctness and efficiency analysis remains the same as before. We only need to argue security.

We show that for every sequence of pairs of $\{(P_0, P_1)\}_\lambda$, where $P_0, P_1 \in \mathcal{P}_\lambda$ have the same functionality, $\hat{P}_0 = i\mathcal{O}(1^\lambda, P_0)$ and $\hat{P}_1 = i\mathcal{O}(1^\lambda, P_1)$ are (sub-exponentially) indistinguishable. We reduce this indistinguishability to the distributional-semantic security of GES. Consider the element distribution $\mathbb{E}(1^\lambda, \mathcal{R})$ that samples $\vec{a}_0 \overset{\$}{\leftarrow} \mathcal{D}_\lambda(\mathcal{R}, P_0)$ and $\vec{a}_1 \overset{\$}{\leftarrow} \mathcal{D}_\lambda(\mathcal{R}, P_1)$ and sets $\vec{z} = \mathsf{null}$; we first show that $\mathbb{E}$ is a $(\vec{\mathbf{v}}, \mathsf{null}, \mathbf{v}_{zt})$-respecting element sampler w.r.t. $\tilde{\mathcal{R}}$.

Towards this, we need to show that, there exists a constant ε, such that, for every ensemble of $((\vec{\mathbf{v}}, \mathsf{null}), \mathbf{v}_{zt})$-respecting arithmetic circuits $\{q_\lambda\}$, it holds that for every $\lambda \in \mathbb{N}$, $q = q_\lambda$, there is a constant c, such that, for every $b \in \{0,1\}$,

$$\Pr[\mathcal{R} \xleftarrow{\$} \tilde{\mathcal{R}}(1^\lambda),\ (\vec{a}_0, \vec{a}_1, \mathsf{null}) \xleftarrow{\$} \mathbb{E}(1^\lambda, \mathcal{R})\ :\ P_{zero}(q(\vec{a}_b)) = c] \geq 1 - \frac{1}{2^{\ell(\lambda)^\varepsilon}}\ , \tag{9}$$

where $\ell(\lambda) = \mathrm{poly}(\log L(\lambda) + \omega(\log \lambda) + t(\lambda))$.

Lemma 7 shows that our scheme $(i\mathcal{O}, \mathsf{Eval})$ satisfies algebraic security, meaning there is a simulators Sim, such that, the following holds w.r.t. $\{q\}_\lambda$, and every $b \in \{0,1\}$,

$$\left|\Pr[\vec{a}_b \xleftarrow{\$} \mathcal{D}_\lambda(\mathcal{R}, P_b)\ :\ P_{zero}(q(\vec{a}_b)) = 1] - \Pr[\mathsf{Sim}^{P_b}(1^\lambda, q) = 1]\right| \leq \mathrm{negl}(\lambda)/2^{t(\lambda)}\ .$$

Furthermore since Sim is deterministic, it implies that

$$\Pr[\vec{a}_b \xleftarrow{\$} \mathcal{D}_\lambda(\mathcal{R}, P_b)\ :\ P_{zero}(q(\vec{a}_b)) = \mathsf{Sim}^{P_b}(1^\lambda, q)] \geq 1 - \mathrm{negl}(\lambda)/2^{t(\lambda)}\ .$$

By the fact that P_0, P_1 are functionally equivalent, it holds that there is $c = \mathsf{Sim}^{P_b}(1^\lambda, q)$ for both $b \in \{0,1\}$.

$$\Pr[\vec{a}_b \xleftarrow{\$} \mathcal{D}_\lambda(\mathcal{R}, P_b)\ :\ P_{zero}(q(\vec{a}_b)) = c] \geq 1 - \mathrm{negl}(\lambda)/2^{t(\lambda)}\ .$$

Finally, since $t(\lambda) >> \log L(\lambda) + \omega(\log \lambda)$, $t(\lambda) = \ell(\lambda)^\varepsilon$ for some constant ε. By construction of $\mathbb{E}$, equation (9) holds and $\mathbb{E}$ is a $(\vec{\mathbf{v}}, \mathsf{null}, \mathbf{v}_{zt})$-respecting element sampler w.r.t. $\tilde{\mathcal{R}}$. Then, it follow from the (sub-exponential) distributional semantic security of GES that encodings of $\vec{a}_0$ and $\vec{a}_1$ under the levels $\vec{\mathbf{v}}$ are (sub-exponential) indistinguishable, which concludes the security of our IO scheme. $\square$

Acknowledgements

The author would like to thank Ran Canetti, Shafi Goldwasser, Shai Halevi, Russell Impagliazzo, Shachar Lovett, Rafael Pass, and Vinod Vaikuntanathan for delightful and interesting discussions. Moreover, the author would like to give special thanks to Benny Applebaum and Stefano Tessaro for many helpful inputs 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.

[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.

[AIK08] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. On pseudorandom generators with linear stretch in nc^0. *Computational Complexity*, 17(1):38–69, 2008.

[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.

[AL15] Benny Applebaum and Shachar Lovett. Algebraic attacks against random local functions and their countermeasures. *Electronic Colloquium on Computational Complexity (ECCC)*, 22:172, 2015.

[Ale03] Michael Alekhnovich. More on average case vs approximation complexity. In *44th Symposium on Foundations of Computer Science (FOCS 2003), 11-14 October 2003, Cambridge, MA, USA, Proceedings*, pages 298–307, 2003.

[App14a] Benny Applebaum. Bootstrapping obfuscators via fast pseudorandom functions. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT 2014, Part II*, volume 8874 of *LNCS*, pages 162–172, Kaoshiung, Taiwan, R.O.C., December 7–11, 2014. Springer, Heidelberg, Germany.

[App14b] Benny Applebaum. *Cryptography in Constant Parallel Time.* Information Security and Cryptography. Springer, 2014.

[BGI$^+$01] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In Joe Kilian, editor, *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, *47th ACM STOC*, pages 439–448, Portland, OR, USA, June 14–17, 2015. ACM Press.

[BPR12] Abhishek Banerjee, Chris Peikert, and Alon Rosen. Pseudorandom functions and lattices. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 719–737, Cambridge, UK, April 15–19, 2012. Springer, Heidelberg, Germany.

[BPW16] Nir Bitansky, Omer Paneth, and Daniel Wichs. Perfect structure on the edge of chaos - trapdoor permutations from indistinguishability obfuscation. In *Theory of Cryptography - 13th International Conference, TCC 2016-A, Tel Aviv, Israel, January 10-13, 2016, Proceedings, Part I*, pages 474–502, 2016.

[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.

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

[BV15a] 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.

[BV15b] Zvika Brakerski and Vinod Vaikuntanathan. Constrained key-homomorphic prfs from standard lattice assumptions - or: How to secretly embed a circuit in your PRF. In *Theory of Cryptography - 12th Theory of Cryptography Conference, TCC 2015, Warsaw, Poland, March 23-25, 2015, Proceedings, Part II*, pages 1–30, 2015.

[BV15c] Zvika Brakerski and Vinod Vaikuntanathan. Constrained key-homomorphic PRFs from standard lattice assumptions - or: How to secretly embed a circuit in your PRF. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015, Part II*, volume 9015 of *LNCS*, pages 1–30, Warsaw, Poland, March 23–25, 2015. Springer, Heidelberg, Germany.

[BV16] Nir Bitansky and Vinod Vaikuntanathan. Indistinguishability obfuscation: From approximate to exact. In *Theory of Cryptography - 13th International Conference, TCC 2016-A, Tel Aviv, Israel, January 10-13, 2016, Proceedings, Part I*, pages 67–95, 2016.

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

[BWZ14] Dan Boneh, David J. Wu, and Joe Zimmerman. Immunizing multilinear maps against zeroizing attacks. Cryptology ePrint Archive, Report 2014/930, 2014. `http://eprint.iacr.org/2014/930`.

[CEMT09] James Cook, Omid Etesami, Rachel Miller, and Luca Trevisan. Goldreich's one-way function candidate and myopic backtracking algorithms. In *TCC*, pages 521–538, 2009.

[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.

[CGP15] Ran Canetti, Shafi Goldwasser, and Oxana Poburinnaya. Adaptively secure two-party computation from indistinguishability obfuscation. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015, Part II*, volume 9015 of *LNCS*, pages 557–585, Warsaw, Poland, March 23–25, 2015. Springer, Heidelberg, Germany.

[CHJV15] Ran Canetti, Justin Holmgren, Abhishek Jain, and Vinod Vaikuntanathan. Succinct garbling and indistinguishability obfuscation for RAM programs. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *47th ACM STOC*, pages 429–437, Portland, OR, USA, June 14–17, 2015. ACM Press.

[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.

[CKP15] Ran Canetti, Yael Tauman Kalai, and Omer Paneth. On obfuscation with random oracles. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015, Part II*, volume 9015 of *LNCS*, pages 456–467, Warsaw, Poland, March 23–25, 2015. Springer, Heidelberg, Germany.

[CLP15] Kai-Min Chung, Huijia Lin, and Rafael Pass. Constant-round concurrent zero-knowledge from indistinguishability obfuscation. In *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I*, pages 287–307, 2015.

[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, *TCC 2015, Part II*, volume 9015 of *LNCS*, pages 468–497, Warsaw, Poland, March 23–25, 2015. Springer, Heidelberg, Germany.

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

[DKR15] Dana Dachman-Soled, Jonathan Katz, and Vanishree Rao. Adaptively secure, universally composable, multiparty computation in constant rounds. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015, Part II*, volume 9015 of *LNCS*, pages 586–613, Warsaw, Poland, March 23–25, 2015. Springer, Heidelberg, Germany.

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

[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 FOCS*, pages 40–49, Berkeley, CA, USA, October 26–29, 2013. IEEE Computer Society Press.

[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.

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

[GHMS14] Craig Gentry, Shai Halevi, Hemanta K. Maji, and Amit Sahai. Zeroizing without zeroes: Cryptanalyzing multilinear maps without encodings of zero. Cryptology ePrint Archive, Report 2014/929, 2014. http://eprint.iacr.org/2014/929.

[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.

[GLSW14] Craig Gentry, Allison Lewko, Amit Sahai, and Brent Waters. Indistinguishability obfuscation from the multilinear subgroup elimination assumption. Cryptology ePrint Archive, Report 2014/309, 2014. http://eprint.iacr.org/2014/309.

[Gol00] Oded Goldreich. Candidate one-way functions based on expander graphs. Cryptology ePrint Archive, Report 2000/063, 2000. http://eprint.iacr.org/2000/063.

[GP15] Sanjam Garg and Antigoni Polychroniadou. Two-round adaptively secure MPC from indistinguishability obfuscation. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015, Part II*, volume 9015 of *LNCS*, pages 614–637, Warsaw, Poland, March 23–25, 2015. Springer, Heidelberg, Germany.

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

[KLW15] Venkata Koppula, Allison Bishop Lewko, and Brent Waters. Indistinguishability obfuscation for turing machines with unbounded memory. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *47th ACM STOC*, pages 419–428, Portland, OR, USA, June 14–17, 2015. ACM Press.

[KMN$^+$14] Ilan Komargodski, Tal Moran, Moni Naor, Rafael Pass, Alon Rosen, and Eylon Yogev. One-way functions and (im)perfect obfuscation. 2014.

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

[LMN89] Nathan Linial, Yishay Mansour, and Noam Nisan. Constant depth circuits, fourier transform, and learnability. In *30th FOCS*, pages 574–579, 1989.

[LPST15] Huijia Lin, Rafael Pass, Karn Seth, and Sidharth Telang. Output-compressing randomized encodings and applications. *IACR Cryptology ePrint Archive*, 2015:720, 2015.

[LPST16] Huijia Lin, Rafael Pass, Karn Seth, and Sidharth Telang. Indistinguishability obfuscation with non-trivial efficiency. In *Public-Key Cryptography - PKC 2016 - 19th IACR International Conference on Practice and Theory in Public-Key Cryptography, Taipei, Taiwan, March 6-9, 2016, Proceedings, Part II*, pages 447–462, 2016.

[MMN15] Mohammad Mahmoody, Ameer Mohammed, and Soheil Nematihaji. More on impossibility of virtual black-box obfuscation in idealized models. Cryptology ePrint Archive, Report 2015/632, 2015. http://eprint.iacr.org/2015/632.

[MMN$^+$16] Mohammad Mahmoody, Ameer Mohammed, Soheil Nematihaji, Rafael Pass, and Abhi Shelat. Lower bounds on assumptions behind indistinguishability obfuscation. In *Theory of Cryptography - 13th International Conference, TCC 2016-A, Tel Aviv, Israel, January 10-13, 2016, Proceedings, Part I*, pages 49–66, 2016.

[MST03] Elchanan Mossel, Amir Shpilka, and Luca Trevisan. On e-biased generators in NC0. In *44th Symposium on Foundations of Computer Science (FOCS 2003), 11-14 October 2003, Cambridge, MA, USA, Proceedings*, pages 136–145, 2003.

[NR95] Moni Naor and Omer Reingold. Synthesizers and their application to the parallel construction of pseudo-random functions. In *36th FOCS*, pages 170–181, Milwaukee, Wisconsin, October 23–25, 1995. IEEE Computer Society Press.

[NR97] Moni Naor and Omer Reingold. Number-theoretic constructions of efficient pseudo-random functions. In *38th FOCS*, pages 458–467, Miami Beach, Florida, October 19–22, 1997. IEEE Computer Society Press.

[NRR00] Moni Naor, Omer Reingold, and Alon Rosen. Pseudo-random functions and factoring (extended abstract). In *32nd ACM STOC*, pages 11–20, Portland, Oregon, USA, May 21–23, 2000. ACM Press.

[NS94] Noam Nisan and Mario Szegedy. On the degree of boolean functions as real polynomials. *Computational Complexity*, 4:301–313, 1994.

[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.

[Pas15] Rafael Pass and abhi shelat. Impossibility of VBB obfuscation with ideal constant-degree graded encodings. Cryptology ePrint Archive, Report 2015/383, 2015. http://eprint.iacr.org/2015/383.

[PST14a] 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.

[PST14b] Rafael Pass, Karn Seth, and Sidharth Telang. Indistinguishability obfuscation from semantically-secure multilinear encodings. In *CRYPTO'14*, 2014.

[SW14a] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In David B. Shmoys, editor, *46th ACM STOC*, pages 475–484, New York, NY, USA, May 31 – June 3, 2014. ACM Press.

[SW14b] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: Deniable encryption, and more. *Proc. of STOC 2014*, 2014.

[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.

8 Proof Sketch of Theorem 4

We briefly sketch why the transformation from compact FE to IO for $\mathsf{P/poly}$ of [AJ15, BV15a] works with compact FE scheme for *bounded size T and bounded input length n*, as long as T and n are sufficiently large and encryption remains significantly faster than T.

We show that when T and n are sufficiently large, in particular, $T > \lambda^{\alpha/\varepsilon}$ for some universal constant α set in the analysis below, the proof follows in two steps.

Step 1: We first follow [BV15a] (BV) and show that IO for $\mathsf{P/poly}$ is implied any compact FE scheme for $T'(\lambda)$-time $n'(\lambda)$-input bit circuits in $\mathsf{P/poly}$, with encryption time bounded by $T'(\lambda)^{1-\varepsilon'}$, for some sufficiently large n' and T'; in particular, time bound satisfies $T'(\lambda) > \lambda^{\alpha'/\varepsilon'}$ w.r.t. a universal constant α' specified below. Let FE' denote such a scheme.

Recall that an indistinguishability obfuscator $i\mathcal{O}$ for $\mathsf{P/poly}$ on input security parameter λ obfuscates circuit C of size bounded by λ (see Definition 4); without loss of generality, we assume that input length of C is $n_C = \lambda$. The BV obfuscator $i\mathcal{O}(\lambda, C)$ proceeds as follows: It generates a fresh instance of FE' for each $i \in \lambda$, $(mpk_i, msk_i) \xleftarrow{\$} \mathsf{FE.Setup}'(1^{\tilde{\lambda}}, T')$, using a larger security parameter $\tilde{\lambda} = \text{poly}(n_C, \lambda) = \text{poly}(\lambda)$[5] and time-bound $T' = T'(\tilde{\lambda})$, and releases a single secret function key fsk_i for each instance. Additionally, it generates two ciphertexts ct_0, ct_1 under the first mpk_1. The functionality of different secret keys are roughly as follows:

- The last secret key fsk_λ is related with a function $f_\lambda(x)$ that evaluates $C(x)$.
- Previous secret keys fsk_i are related with a f_i that on input an i-bit prefix $x_{\leq i}$ generates FE encryptions of two possible continuations $x_{\leq i}||0$ and $x_{\leq i}||1$, using pseudo-random coins generated via a PPRF with key K^i.

To compute $C(x)$, the evaluator chooses $ct^1 = ct_{x_1}$ as the encryption of the first bit of x; then it iteratively performs FE evaluation to obtain encryption of all prefixes: In iteration $i < \lambda$, it evaluates ciphertext ct^i of prefix $x_{\leq i}$ with fsk_i to obtain ciphertexts ct_0^{i+1}, ct_1^{i+1} of extensions

[5]The security parameter is scaled up for complexity leveraging.

$x_{\leq i}||0, x_{\leq i}||1$, and chooses the x_{i+1}^{th} ciphertext for the next iteration. In iteration λ, the output $C(x)$ is obtained by evaluating the final ciphertext ct^{λ} of x with the last secret key fsk_{λ}.

For security analysis, the actual functions associated with fsk_i's involves much more than just evaluating C and $\mathsf{FE.Enc}'$. (We refer the reader to [BV15a] for details.) Here, we are concerned with the input length n_i and size $|f_i|$ of the function f_i being computed by FE'. By the analysis of [BV15a],

$$n_i \leq n_C + \tilde{\lambda} + 1 \leq 2\tilde{\lambda}.$$

If the input length bound n' of FE' is sufficiently large, that is, $n'(\tilde{\lambda}) \geq 2\tilde{\lambda}$, FE' supports encrypting all inputs used in [BV15a]. As a result, we can also assume without loss of generality that all inputs are padded to exactly length n' when encrypted.

Furthermore, by the analysis of [BV15a], the sizes of function f_i's are bounded by

$$|f_i| \leq \tilde{\lambda}^{\alpha_1} \max(|C|, T_{\mathsf{FE.Enc}'}) ,$$

where $\tilde{\lambda}^{\alpha_1}$ is a fixed multiplicative polynomial overhead, and $T_{\mathsf{FE.Enc}'}$ is the worst-case time for encrypting a message of length n' under mpk_i produced using $\tilde{\lambda}$ and T'. By the fact that encryption time of FE' satisfies $T_{\mathsf{FE.Enc}'} \leq T'(\tilde{\lambda})^{1-\varepsilon'}$. For the construction to be well-defined, it must hold that the sizes of the functions computed do not exceed the time bound, i.e., $|f_i| \leq T'(\tilde{\lambda})$. Since T' satisfies $T'(\tilde{\lambda}) \geq \tilde{\lambda}^{\alpha'/\varepsilon'}$ and $|C| \leq \lambda$, we have

$$|f_i| \leq \tilde{\lambda}^{\alpha_1} \max(|C|, T_{\mathsf{FE.Enc}'}) \leq \tilde{\lambda}^{\alpha_1} {T'}^{1-\varepsilon'} \leq \tilde{\lambda}^{\alpha'} {T'}^{1-\varepsilon'} \leq T' .$$

The second last inequality holds if the universal constant α' associated with FE' is no smaller than α_1. Therefore, though FE' does not support arbitrary polynomial computation, the IO construction is still well-defined, and the analysis in [BV15a] applies identically.

Step 2: Following [ABSV15] (ABSV), we show that a FE scheme FE' for $\mathsf{P/poly}$ as required in Step 1, can be constructed from any compact FE scheme FE for $T(\lambda)$-time $n(\lambda)$-input bit circuits in NC^1 with encryption time bounded by $T(\lambda)^{1-\varepsilon}$, for sufficiently large n and T; in particular, the time bound satisfies $T(\lambda) \geq \lambda^{\alpha/\varepsilon}$ for some universal constant α specified below, assuming a weak PRF scheme wPRF in NC^1.

We sketch the ABSV construction: The secret key $sk'_{C'}$ of FE' for a circuit $C' \in \mathsf{P/poly}$ is a secret key sk_C of FE for a related circuit $C \in \mathsf{NC}^1$. C has a hardwired tag $\tau \xleftarrow{\$} \{0,1\}^{\lambda}$ and ciphertext ct encrypted under a symmetric encryption scheme with secret key sk; on input (x, K, sk, b), C either outputs a randomized encoding $\mathsf{RE}(1^{\lambda}, C', x)$ using pseudo-random coins $\mathsf{wPRF}(K, \tau)$ if $b = 0$, or simply decrypts the ciphertext $\mathsf{Dec}(sk, ct)$ if $b = 1$. Ciphertexts of FE' correspondingly encrypt tuple of form (x, K, sk, b).

By the analysis of [ABSV15], the relations between the input lengths n_C and $n_{C'}$ of C and C', and that between their sizes are: $n_C = n_{C'} + 2\lambda + 1$ and $|C| \leq \lambda^{\alpha_2}|C'|$, where λ^{α_2} is a fixed multiplicative polynomial overhead. Therefore, the resulting FE' can compute circuits of bounded input length and bounded size

$$n'(\lambda) = n(\lambda) - 2\lambda - 1 , \qquad T'(\lambda) = T(\lambda)/\lambda^{\alpha_2} .$$

By the $(1-\varepsilon)$-compactness of FE, time for encryption $T_{\mathsf{FE.Enc}'}$ is bounded by $T^{1-\varepsilon}$. Hence,

$$T_{\mathsf{FE.Enc}'} \leq T^{1-\varepsilon} = \frac{T^{1-\varepsilon/2}}{T^{\varepsilon/2}} \leq \left(\frac{T}{T^{\varepsilon/2}}\right)^{1-\varepsilon/2} \leq \left(\frac{T}{\lambda^{\alpha_2}}\right)^{1-\varepsilon/2} = {T'}^{1-\varepsilon/2}$$

Since $T \geq \lambda^{\alpha/\varepsilon}$ for a universal constant α, the last inequality holds if $\alpha > 2\alpha_2$. Therefore FE' is $(1-\varepsilon')$-compact for $\varepsilon' = \varepsilon/2$.

Finally, we show that T' is sufficiently large $T' > \lambda^{\alpha'/\varepsilon'}$, this is because

$$T' = \frac{T}{\lambda^{\alpha_2}} > \lambda^{\alpha/\varepsilon - \alpha_2} > \lambda^{(\alpha-\alpha_2)/2\varepsilon'} \geq \lambda^{\alpha'/\varepsilon'}$$

where the last inequality holds when the universal constant $\alpha > 2\alpha' + \alpha_2$.

Combining the above analysis, when $\alpha = 2(\alpha' + \alpha_2)$, the resulting FE' satisfies all requirements from Step 1, which concludes the proof.