

Indistinguishability Obfuscation from Constant-Degree Graded Encoding Schemes

Huijia Lin^(✉)

University of California, Santa Barbara, USA
rachel.lin@cs.ucsb.edu

Abstract. We construct an indistinguishability obfuscation (IO) scheme for all polynomial-size circuits from *constant-degree* graded encoding schemes, assuming the existence of a subexponentially secure pseudo-random generator computable by constant-degree arithmetic circuits, and the subexponential hardness of the Learning With Errors (LWE) problems. Previously, all candidate general purpose IO schemes rely on polynomial-degree graded encoding schemes.

1 Introduction

Program obfuscation [13] 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. [36]. 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 [46], trapdoor permutations [15], public-key encryption [60], to fully homomorphic encryption [26], but also in reaching new possibilities, from functional encryption [36], 2-round adaptively secure multi-party computation protocols [24, 34, 37], succinct garbling in time independent of the computation time [14, 25, 47, 48], to constant-round concurrent ZK protocol [28]. 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 [4, 7, 12, 22, 40, 59, 62]. 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 et al. [35], 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 P/poly , assuming the LWE assumption [36].

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.

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 [31, 32, 35, 38] are susceptible to the so called “zeroizing attacks” [20, 27, 30, 35, 39] 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 constant-degree graded encoding schemes. In comparison, previous IO schemes rely on polynomial degree graded encodings, polynomial in the size of the obfuscated circuit. This improvement is established exactly via a stronger bootstrapping theorem.

- *Bootstrapping IO for Constant-Degree Arithmetic Circuits.* We show that there is a class \mathcal{C} of special-purpose constant-degree arithmetic circuits (i.e., corresponding to constant-degree polynomials), such that, special-purpose IO for \mathcal{C} can be bootstrapped to general purpose IO for $\mathsf{P/poly}$, assuming the sub-exponential hardness of LWE, and the existence of a sub-exponentially secure Pseudo-Random Generator (PRG) computable by constant-degree arithmetic circuits. An candidate of such a PRG is Goldreich’s PRG in NC^0 [41].
- *Constant-Degree Graded Encodings Suffice.* Then, we show that special purpose IO for \mathcal{C} can be constructed from constant-degree graded encoding schemes.

Relation with Recent Works [16, 51, 58]. At a first glance, our main theorem is surprising in light of the recent results by [17, 51, 58]. 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 [17, 51, 58] to eliminate the use of graded encodings.

Moreover, it is interesting to note that our construction of IO for P/poly uses as a component the transformation from sub-exponentially secure compact functional encryption to general purpose IO by [3, 16]. 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. 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 [17, 51, 58], 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 P/poly , there is an IO scheme for 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. [59]), and obtain general-purpose IO via bootstrapping. Below, we explain each part in more detail.

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 [36], 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 [26] is based on Applebaum’s idea of bootstrapping VBB for NC^0 [5], 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) [61] computable in NC^1 [23].

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” for 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, IO for $\{C_\lambda\}$ with universal efficiency (explained below) can be bootstrapped into IO for P/poly .*

Let us explain our bootstrapping theorem in more detail.

Model of Constant Degree Computations. 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 $f : \{0, 1\}^n \rightarrow \{0, 1\}^m$, 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). However, 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 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 any ring \mathcal{R} .

This model of constant-degree computations is quite weak, in fact, so weak that it is equivalent to NC^0 . Nisan and Szegedy [56] 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 .

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 $\mathbb{D} \in \{0, 1\}^n$ (i.e., we only care about inputs in \mathbb{D}).

- *Constant-Degree Computations with Partial Domains:* We say that a Boolean function f (or logic circuit C) with domain $\mathbb{D} \in \{0, 1\}^n$ 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} .

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 even statistical IO is feasible for such circuits.

Trivial, Statistical IO for Constant Degree Computations. Let C be a logic circuit computable by a degree- d arithmetic circuit AC , which corresponds to a degree- d polynomial. At a high-level, because degree- d polynomials can be learned in $\text{poly}(n^d)$ 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 $x_1, \dots, x_\ell \in \{0, 1\}^n$, such that, their corresponding monomial values X_1, \dots, X_ℓ are linearly independent. Now, the obfuscation \tilde{C} of C is simply the set of input output pairs $(x_1, y_1), \dots, (x_\ell, y_\ell)$ where $y_i = C(x_i)$.

Given \tilde{C} , we can to evaluate C on any input x , since $C(x) = L(x)$ over any ring, 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 only 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.

Computational 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, $h(d) = \log \log \log(d)$. For simplicity, we consider the above universal efficiency.

General Connection between Complexity of PRG and Complexity of Seed Class. Finally, we note that our bootstrapping theorem can be generalized to establish a connection between the complexity of PRG and the complexity of “seed classes” sufficient for bootstrapping IO. Consider any PRG scheme PRG (not necessarily computable in constant degree). There is a family $\{\mathcal{C}_\lambda\}$ of special-purpose

oracle circuits that have constant degree (such a circuit can be computed by an arithmetic circuit with oracle gates, and its degree is the degree of the arithmetic circuit when replacing the oracle gates with additions), such that, IO for the class of composed circuits in $\{\mathcal{C}_\lambda^{\text{PRG}}\}$, with again universal efficiency, can be bootstrapped into IO for P/poly .

Proposition 2 (Informal, General Bootstrapping Theorem). *Assume a PRG scheme PRG and LWE with sub-exponential hardness. There is a class of special-purpose oracle circuits $\{\mathcal{C}_\lambda\}$ that have constant degree, such that, special purpose IO for $\{\mathcal{C}_\lambda^{\text{PRG}}\}$ with universal efficiency can be bootstrapped into IO for P/poly .*

In particular, plugging in a constant-degree PRG yields Proposition 1, and plugging in a PRG in AC^0 or TC^0 establishes that IO for AC^0 or TC^0 with universal efficiency suffices for constructing general purpose IO.

Given the general connection between the complexity of PRG and that of the seed class, we summarize the state-of-the-art of low depth PRG at the end of the introduction.

Techniques. Our starting point is two beautiful recent works by Bitansky and Vaikuntanathan [16] and Ananth and Jain [3] showing that sub-exponentially secure (sublinearly) compact FE for NC^1 implies IO for P/poly . Unfortunately, so far, the former is only known from IO for NC^1 ; thus, our goal is to construct compact FE using IO for the simplest circuits.

Technically, the transformation in the first step is similar to that in [3, 49]. However, the former [3] requires IO for a class of special-purpose Turing machines (as opposed circuits). Our transformation uses the same idea as in [49], but requires a much more refined analysis in order to identify and simplify the circuits, whose special structure plays a key role later.

The work of Ananth and Jain [3], and another very recent work by the author, Pass et al. [49] 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 [43]) can be transformed into a compact FE for all NC^1 circuits with multiple output bits, using IO for circuits (Turing machines in [3]) with only a logarithmic number $c \log \lambda$ of input wires; such circuits have λ^c -sized truth table. [49] further weakens the efficiency requirement on such IO schemes: As long as the IO scheme outputs obfuscated circuits whose size is sub-linear in the size of the truth table (matching the sub-linear compactness of FE), the transformation goes through.

However, the circuits used in [3, 49] are complex, in NC^1 . In this work, we significantly reduce the complexity of the circuits using more refined analysis and a number of new techniques. For example, 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 $\mathbb{D} \subset \{0, 1\}^{\text{poly}(\lambda)}$. 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 $i, i' \in [\text{poly}(\lambda)]$ 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.

Part 2: Special Purpose IO in Constant-Degree Ideal Graded Encoding Model. Ideal grade encoding model [7, 12, 22, 35, 62] 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 [51, 58], 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) [7, 12, 22, 62]. The dilemma is that such models are so powerful that even general purpose VBB obfuscation is feasible, which is impossible in the plain model [13]. Two recent works [51, 58] 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 for the simple class of constant-degree computations, it is sufficient.

Proposition 3 (Informal, Special-Purpose IO in Ideal Model). *There is a universally efficient IO scheme for the class $\{C_\lambda\}$ of constant-degree special-purpose circuits in Proposition 1, 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 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 [7] 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 4.

Part 3: Instantiation with Concrete Graded Encoding Schemes. The final part combine our bootstrapping theorem (Proposition 1) with our special-purpose IO scheme (Proposition 4) 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 et al. [59] 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 et al. [40] 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 [59], we show that our special purpose IO schemes in Proposition 4 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 4 (Informal, Special-Purpose IO in the Plain Model). *There is a universally efficient IO scheme for the class $\{C_\lambda\}$ of constant-degree special-purpose circuits in Proposition 1, assuming 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).

1.2 Low Depth PRG

We survey constructions of low depth PRGs. Some of the texts below are taken verbatim from Applebaum’s book [6].

The existence of PRG in TC^0 follows from a variety of hardness assumption including intractability of factoring, discrete logarithm, or lattice problems

¹ We note that the security of (variants of) our IO scheme could potentially be proven from the multilinear subgroup elimination assumption of [40]; we leave this as future work.

(e.g. [11, 53–55]). 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_4^0 (with output locality 4) achieving super-linear stretch [33, 52]. On the positive side, Applebaum et al. [8] 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 [9], previously conjectured by Alekhnovich [1]. Unfortunately, to the best of our knowledge, there is no construction of PRG in NC^0 (or even AC^0) with polynomial stretch from well-known assumptions. But, candidate construction exists.

Goldreich’s Candidate PRGs in NC^0 . Goldreich’s one-way functions $f : \{0, 1\}^n \rightarrow \{0, 1\}^m$ where each bit of output is a fixed predicate P of a constant number d of input bits, chosen at random or specified by a bipartite expander graph with the right degree, 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 successfully attacks when the choice of the predicate P avoids certain degenerating cases [10, 18, 29, 57]. Notably, O’Donnell and Witmer [57] gave evidence for the security of Goldreich’s PRGs with polynomial 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 Sects. 3, 4, and 5.

In Sect. 2, we formalize our model of constant-degree computations, IO with universal efficiency, and provide basic preliminaries. In Sect. 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 Sect. 4, we show that the class of special purpose circuits identified in Sect. 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 Sect. 5. Due to the lack of space, we refer the readers to the full version on how to instantiate our special-purpose IO with semantically secure graded encodings.

2 Preliminaries

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

Due to the lack of space, we omit definitions of standard cryptographic primitives such as, PRG, PPRF, (compact) functional encryption and randomized encodings (see [3, 8, 16]), and only discuss our models of computation and give definitions of IO and universal efficiency below.

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 $f : \mathbb{D} \rightarrow \{0, 1\}^m$ 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}_λ is associated with domain \mathbb{D}_λ . We use the shorthands $\{\mathcal{C}_\lambda\}$ and $\{\mathbb{D}_\lambda\}$.
- A family of circuit classes $\{\{\mathcal{C}_\lambda^x\}\}^{x \in X}$ is a set of circuit classes, where each circuit class $\{\mathcal{C}_\lambda^x\}$ 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 $\{\{\mathcal{C}_\lambda^x\}\}$. A family of circuit classes can also be associated with domains $\{\{\mathbb{D}_\lambda^x\}\}$, meaning that each set \mathcal{C}_λ^x is associated with domain \mathbb{D}_λ^x . For example, NC^1 circuits can be described as a family of circuit classes $\{\{\mathcal{C}_\lambda^d\}\}^{d \in \mathbb{N}}$, where for every $d \in \mathbb{N}$, the circuit class $\{\mathcal{C}_\lambda^d\}$ 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}_λ with domain \mathbb{D}_λ .

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}_λ 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 $\{\{\mathcal{C}_\lambda^x\}\}$ with domains $\{\{\mathbb{D}_\lambda^x\}\}$ has constant degree, if for every $x \in X$, circuit class $\{\mathcal{C}_\lambda^x\}$ with domains $\{\mathbb{D}_\lambda^x\}$ 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 polynomial-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.

2.2 Indistinguishability Obfuscation

We recall the notion of indistinguishability obfuscation for a class of circuit defined by [13], adding the new dimension that the class of circuits may have restricted domains $\{\mathbb{D}_\lambda\}$.

Definition 1 (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 $\{C_\lambda\}_{\lambda \in \mathbb{N}}$ (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}_λ), 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}_λ), the following ensembles of distributions are μ -indistinguishable,*

$$\begin{aligned} &\{C_{1,\lambda}, C_{2,\lambda}, i\mathcal{O}(1^\lambda, C_{1,\lambda})\}_\lambda \\ &\{C_{1,\lambda}, C_{2,\lambda}, i\mathcal{O}(1^\lambda, C_{2,\lambda})\}_\lambda \end{aligned}$$

In the above definition, μ can be either negligible for standard IO, or subexponentially small for subexponentially secure IO.

Definition 2 (IO for P/poly). *A uniform PPT machine $i\mathcal{O}_{\text{P/poly}}(\star, \star)$ is an indistinguishability obfuscator for P/poly if it is an indistinguishability obfuscator for the class $\{C_\lambda\}$ of circuits of size at most λ .*

2.3 Indistinguishability Obfuscation for Families of Circuit Classes

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

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.

We consider the following notion of universal efficiency in this work.

Definition 4 (Universal Efficiency). A family of indistinguishability obfuscators $\{i\mathcal{O}^x\}_{x \in X}$ for a family of circuit class $\{\{\mathcal{C}_\lambda^x\}\}_{x \in X}$ (with potentially restricted domains $\{\{\mathbb{D}_\lambda^x\}\}$) 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 $C \in \mathcal{C}_\lambda^x$.

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 .

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 Fig. 1, such that, the following holds:

- If there exists a family $\{i\mathcal{O}^{T,n}\}$ of 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 an IO scheme for P/poly .

We note in Sect. 3.1 that all the underlying primitives of the above Proposition are implied by the sub-exp hardness of LWE.

Overview. Towards the proposition, recall that recent works [2, 3, 16] show that to construct IO for P/poly , it suffices to construct a compact FE scheme for NC^1 circuits. Formally,

Theorem 2. [2, 3, 16] 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 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 $\text{NC}^{1,T,n}$ be the subclass of NC^1 circuits with at most size $T(\lambda)$ and at most $n(\lambda)$ input bits.

- Our first step (in Sect. 3.2) constructs an FE scheme $\text{FE}^{T,n}$ for $\text{NC}^{1,T,n}$ from any 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:

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

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 Sect. 3.3) 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)

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

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 [2, 3, 16] to go through. More specifically, the final IO scheme for 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 [3, 49]. However, the former [3] requires IO for a class of special-purpose Turing machines (as opposed circuits). Our transformation uses the same idea as in [49], but requires a much more refined analysis in order to identify and simplify the circuits, whose special structure plays a key role later.

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 et al. [43]: 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 3 (Compact FE scheme for Boolean NC^1 Circuits [43]). *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 [19, 21, 45]; but, the evaluation algorithms of these PPRF schemes have high depth. Recently, Brakerski and Vaikuntanathan [23] 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 [8, 44], which is in turn implied by sub-exp secure LWE.

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

Constant: A security parameter $\lambda \in \mathbb{N}$, a time bound $T \in \mathbb{N}$, a threshold $i^* \in \{0, \dots, T+1\}$, 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{H} .

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

Procedure:

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

Output: Encoding \hat{H}_i .

Padding: The hardwired encoding \hat{H} 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 .

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

Fig. 1. Special-Purpose circuit P

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

Fix arbitrary polynomials T and n . We present an FE scheme $\text{FE}^{T,n}$ for $\text{NC}^{1,T,n}$ from IO for $\{\mathcal{P}_\lambda^{T,n}\}$. Our construction starts with a compact FE scheme for Boolean NC^1 circuits $\text{bFE} = (\text{bFE.Setup}, \text{bFE.Enc}, \text{bFE.Dec})$ (as discussed in

Sect. 3.1, such a scheme can be constructed from LWE), and transforms it into $\text{FE}^{T,n}$. The transformation makes use of the following additional building blocks:

- a puncturable PRF $\text{PPRF} = (\text{PRF.Gen}, \text{PRF.Punc}, \text{F})$ for input domain $\{0, 1\}^\lambda$.
- a randomized encoding scheme $\text{RE} = (\text{RE.Enc}, \text{RE.Eval})$ 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 $P[\lambda, T, n, \text{mpk}, i^*, K, m_1, y, m_0]$ defined in Fig. 1.

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

Construction of $\text{FE}^{T,n}$. For any $\lambda, T = T(\lambda)$ 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 $\text{FE}^{T,n} = (\text{FE.Setup}, \text{FE.KeyGen}, \text{FE.Enc}, \text{FE.Dec})$ proceeds as follow:

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

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

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

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

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

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

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 $c_i = \text{RE.Eval}(\hat{\Pi}_i)$.
3. For $i \in [T]$, evaluate c_i with sk to obtain $y_i = \text{bFE.Dec}(sk, c_i)$.
4. Output $y = y_1 || \dots || y_T$.

It is clear that all algorithms above are PPT. Below, we first analyze the encryption efficiency of $\text{FE}^{T,n}$ in Lemma 1 and then show its correctness and security in Lemma 2.

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

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

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

- It follows from the efficiency of PPRF that Step 1 of 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 Fig. 1).
- It follows from the efficiency of $i\mathcal{O}^{T,n}$ that the second step of encryption takes time $\text{Time}_{i\mathcal{O}^{T,n}}(1^\lambda, P) = p^{T,n}(\lambda, \eta(\lambda, n, \log T))$.

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

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.*

We omit the of this lemma due to the lack of space.

3.3 Obtaining IO for P/poly

By the construction of FE scheme $\text{FE}^{T,n}$ for $\text{NC}^{1,T,n}$ in Sect. 3.2, 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,*

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

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

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

Clearly, this family of FE schemes $\{\text{FE}^{T,n}\}$ gives a compact FE scheme for $\text{NC}^1 = \{\text{NC}^{1,T,n}\}$, and hence already implies IO for P/poly by Theorem 2 shown in [2, 3, 16]. 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 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 [2, 3, 16].

Theorem 4 (Refined Version of Theorem 2, Implicit in [2, 3, 16]). *Assume the existence of a sub-exponentially secure weak PRF in NC^1 , and a (single-query, public-key) FE scheme for $\text{NC}^{1,T,n}$, with encryption time bounded by $T(\lambda)^{1-\varepsilon}$, for sufficiently large polynomials n and T . Then, there exists an indistinguishability obfuscator for 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 P/poly. This concludes Proposition 5.

4 Special-Purpose Circuits in Constant Degree

Assuming a constant-degree PRG, we show how to implement the special-purpose circuits in Fig. 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 Fig. 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.

By Proposition 5, and the fact that all underlying primitives of the Proposition are implied by the hardness of LWE (see the discussion in Sect. 3.1), we obtain the following bootstrapping theorem.

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 P/poly.*

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 $\text{poly}(\lambda, n, \log T)$.

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 $i^* \in \{0, \dots, T+1\}$, 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 [23] 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 [50].

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 $\{0, \dots, T+1\}$ (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 $i^* \in \{0, \dots, T+1\}$, using an alphabet Σ consisting of a polynomial number of vectors of length λ ,

$$\Sigma = \{\mathbf{e}_0, \dots, \mathbf{e}_\lambda\}, \quad (1)$$

where \mathbf{e}_j for $j \in \{0, \dots, \lambda\}$ 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 $\lceil \log(\lambda+1) \rceil$, denoted as $i = i_1 || i_2 || \dots || i_c$. Therefore, using alphabet Σ ,

$$i \stackrel{\Sigma}{=} \mathbf{e}_{i_1} || \dots || \mathbf{e}_{i_c}, \text{ with length } |i|_\Sigma = c\lambda,$$

where $a \stackrel{\Sigma}{=} b$ 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 $b_{i_k,=}^k$ is set to 1 iff $i_k = i_k^*$. With the new representation, this equality can be tested by simply computing $b_{i_k,=}^k = \mathbf{e}_{i_k} \cdot \mathbf{e}_{i_k^*}^k$ 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 $b_{i,=} = \prod_{k \in [c]} b_{i_k,=}^k$ in constant degree c . Testing other relations, smaller than and greater than, between i and i^* can be performed similarly. See Sect. 4.1 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 polynomial stretch. The most natural idea is to construct a PPRF using the GGM PRF tree [42] as done in previous constructions of PPRF [19, 21, 45]. 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 $\text{mux}(\mathbf{v}, j) = \mathbf{v}_j$, which has at least depth $\Omega(\log |\mathbf{v}|)$ when j is represented in binary. But, again thanks to our new input presentation $j \stackrel{\Sigma}{=} \mathbf{e}_j$, mux can be implemented as $\mathbf{v} \cdot \mathbf{e}_j$ in degree 2. See Sect. 4.2 for details on the PPRF.

Finally, we put all pieces together in Sect. 4.3. Our final implementation of special purpose circuits had degree of order $\exp(\log_\lambda(Tn))$.

4.1 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 $b_{i,<}^k$, $b_{i,=}^k$, $b_{i,>}^k$ 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_k^* , i_k^* ; they can be computed as follows:

- $b_{i,=}^k$ can be computed as the inner product $b_{i,=}^k = \mathbf{e}_{i_k} \cdot \mathbf{e}_{i_k^*}$.
- $b_{i,<}^k$ can be computed as the inner product $b_{i,<}^k = \mathbf{e}_{i_k} \cdot \mathbf{e}_{<i_k^*}$, where $\mathbf{e}_{<i_k^*}$ denote the vector that contains 1s in the first $i_k^* - 1$ positions, and 0s in the rest.
- $b_{i,>}^k$ can be similarly computed as the inner product $b_{i,>}^k = \mathbf{e}_{i_k} \cdot \mathbf{e}_{>i_k^*}$, where $\mathbf{e}_{>i_k^*}$ 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 Fig. 2.

4.2 PRF Evaluation in Constant-Degree

The special purpose circuits require a PPRF function with input domain $\{0, \dots, T\}$, 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 $\{0, \dots, D(\lambda)\}$, 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.*

Performing Comparisons Compare(i)

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

Input: an input $i \in [T]$ represented as $i \stackrel{\Sigma}{=} (\mathbf{e}_{i_k})_{k \in [c]}$.

Procedure:

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

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

$$b_{i,=} = \prod_{k \in [c]} b_{i,=}^k. \quad (2)$$

Testing $i < i^*$ 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,

$$b_{i,<} = 1 - \prod_{k \in [c]} \left(1 - \left(\prod_{j < k \in [c]} b_{i,=}^j \right) \times b_{i,<}^k \right). \quad (3)$$

Testing $i > i^*$ 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,

$$b_{i,>} = 1 - \prod_{k \in [c]} \left(1 - \left(\prod_{j < k \in [c]} b_{i,=}^j \right) \times b_{i,>}^k \right). \quad (4)$$

Fig. 2. Performing comparisons between i and i^* in constant degree.

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, $\text{qPRG}(s) = \text{PRG}^I(s)$. 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 $\mathbf{v} = \mathbf{v}[1], \dots, \mathbf{v}[\lambda]$ 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.

Using qPRG, we now construct a PPRF scheme $\text{PPRF} = (\text{PRF.Gen}, \text{PRF.Punc}, \text{F})$ 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 $\{0, \dots, D\}$ represents inputs under alphabet Σ (in Eq. (1)), or alternatively, the input domain is Σ^c .

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

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

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

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

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

Efficiency and security: The only difference between the above scheme and the original constructions of PPRF based on GGM tree [19, 21, 45] 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 [19, 21, 45] goes through even when the tree has higher degree; we omit details here.

PRF Evaluation $F(K(-i^*), i)$

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

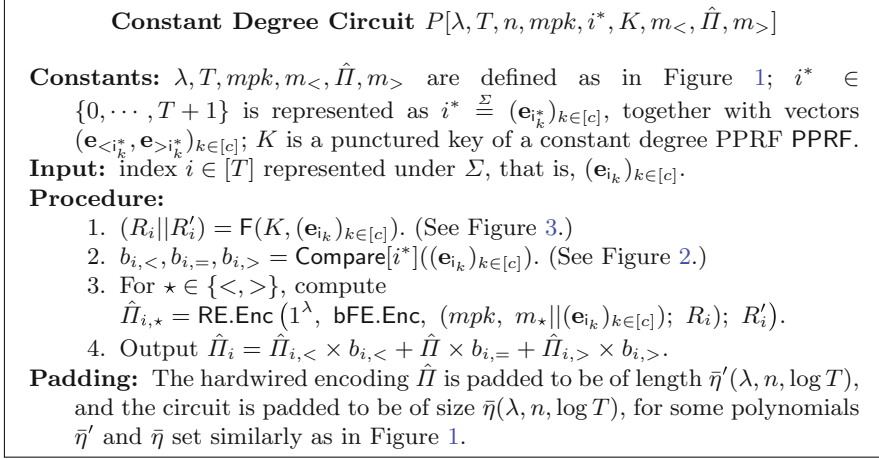
Procedure:

1. For every $k \in [c]$, compute $b_{i,=}^k = \mathbf{e}_{i_k^*} \cdot \mathbf{e}_{i_k^*}$, which indicates whether the k^{th} blocks i_k^* and i_k are equal.
2. For every $k \in [c]$, compute d_i^k indicating whether the following occurs: The first $k-1$ blocks of i and i^* are equal, but the k^{th} block differs.

$$d_i^k = \left(\prod_{j < k \in [c]} b_{i,=}^j \right) \times \left(1 - b_{i,=}^k \right).$$
3. For every $k \in [c]$, do:
 - Select the i_k^{th} element in $\mathbf{v}_k[\neq i_k^*]$, $K_k^k = \mathbf{v}_k[\neq i_k^*] \cdot \mathbf{e}_{i_k}$.
 - For $j = k+1$ to c , compute $\mathbf{w}_j = \text{qPRG}(K_{j-1}^k)$, $K_j^k = \mathbf{w}_j \cdot \mathbf{e}_{i_j}$.
4. Compute the final output $y = \Sigma_{k \in [c]} (K_c^k \times d_i^k)$.

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.

Fig. 3. Constant-degree PRF evaluation

Fig. 4. Special-purpose circuit P in constant degree

4.3 Putting Pieces Together

Given the sub-routine **Compare** and a constant-degree PPRF scheme **PPRF** with domain $\{0, \dots, T+1\}$ and appropriate output length $L(\lambda) = \text{poly}(\lambda, n, \log T)$, a constant-degree implementation the special-purpose circuits is presented in Fig. 4, where Step 1 and 2 evaluate the new functions **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 $(\mathbf{e}_{i_1}, \dots, \mathbf{e}_{i_c})$. 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 $\text{poly}(\lambda)$, and so is the evaluation of function **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 $\text{poly}(\lambda, n, \log T)$. Therefore, the worst-case run-time and hence circuit size is bounded by $u(\lambda, n, \log T)$, for some universal polynomial u .

5 IO for Special-Purpose Circuits in Ideal Model

In this section, we construct IO for our special-purpose circuits in ideal graded encoding model. Due to the lack of space, we provide only an overview of our construction. We refer the reader to the full version for more details.

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 [12, 22, 40, 59] or directly obfuscate circuits [4, 7, 62]. 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 [7] (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” [12], 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 $(r, rw) \xleftarrow{\$} \text{EG}(w)$ with a random r . Encodings of El-Gamal encodings of w_1 and w_2 , $(R_1 = [r_1]_{\mathbf{v}_1}, Z_1 = [r_1 w_1]_{\mathbf{v}_1})$ and $(R_2 = [r_2]_{\mathbf{v}_2}, Z_2 = [r_2 w_2]_{\mathbf{v}_2})$ can be “added” using an addition gadget \oplus that does $(R_1 R_2 = [r_1 r_2]_{\mathbf{v}_1 + \mathbf{v}_2}, Z_1 R_2 + Z_2 R_1 = [r_1 r_2 (w_1 + w_2)]_{\mathbf{v}_1 + \mathbf{v}_2})$, 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 $\bar{y} = U(y_1, \dots, y_{n+m})$. 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 $\text{EG}(P_i)$, encode $\text{EG}(w_{n+i})$ 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, \dots, y_{n+m} . An honest evaluation yields encodings of $\text{EG}((U(x, P), \bar{y}))$. By additionally releasing encodings of $\text{EG}((1, \bar{y}))$, 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 $\text{EG}(Y(x, P), Y(y_1, \dots, y_{n+m}))$ with some $Y \neq U$, and the value $Y(y_1, \dots, y_{n+m})$ 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- $(n2^{\text{depth}})$ 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, \dots, 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 $Z_s^i = [s]_{\mathbf{v}_s^i}$, and \hat{Z}_s^i of a random value under set $\hat{\mathbf{v}}_s^i$. Consider a simplified example: Set $\mathbf{v}_s^i = (0 \cdots 0, 1, 0 \cdots 0, 1)$ with 1 at position s and $\lambda + 1$, and $\hat{\mathbf{v}}_s^i = (d \cdots d, 0, d \cdots d, 0)$ correspondingly. (As in Point 1 above,) the only way to reach (d, \dots, d) is using Z_s^i for some s consistently followed by a multiplication with \hat{Z}_s^i . The actual encoding is more complicated as s is described as a λ -bit string s_1, \dots, s_λ , and each bit needs to be encoded separately $\mathbf{Z}_s^i = \{[s_j]_{\mathbf{v}_s^i}\}_j$.
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 \oplus . Fortunately, thanks to the special domain Σ^c , encodings for different bits of an input symbol \mathbf{Z}_s^i have the same level \mathbf{v}_s^i . To allow for using \oplus , we further let their El-Gamal encodings share the same randomness r_s^i , that is, $R_s^i = [r_s^i]_{\mathbf{v}_s^i}$ and $\mathbf{Z}_s^i = \{[r_s^i s_j]_{\mathbf{v}_s^i}\}_j$. 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 \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, $\hat{\oplus}$ 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 [7] as opposed to the simple obfuscator described above; like the robust obfuscator of [7], 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, [30] showed that a simplified version of the simple obfuscator of [7] 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 .

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

References

1. Alekhnovich, M.: More on average case vs approximation complexity. In: Proceedings of the 44th Symposium on Foundations of Computer Science (FOCS 2003), 11–14 October 2003, Cambridge, MA, USA, pp. 298–307 (2003)
2. Ananth, P., Brakerski, Z., Segev, G., Vaikuntanathan, V.: From selective to adaptive security in functional encryption. In: Gennaro, R., Robshaw, M. (eds.) CRYPTO 2015, Part II. LNCS, vol. 9216, pp. 657–677. Springer, Heidelberg (2015)
3. Ananth, P., Jain, A.: Indistinguishability obfuscation from compact functional encryption. In: Gennaro, R., Robshaw, M.J.B. (eds.) CRYPTO 2015, Part I. LNCS, vol. 9215, pp. 308–326. Springer, Heidelberg (2015)
4. Ananth, P.V., Gupta, D., Ishai, Y., Sahai, A.: Optimizing obfuscation: avoiding Barrington’s theorem. In: Ahn, G.-J., Yung, M., Li, N. (eds.) ACM CCS 2014, Scottsdale, AZ, USA, 3–7 November 2014, pp. 646–658. ACM Press (2014)

5. Applebaum, B.: Bootstrapping obfuscators via fast pseudorandom functions. In: Sarkar, P., Iwata, T. (eds.) ASIACRYPT 2014, Part II. LNCS, vol. 8874, pp. 162–172. Springer, Heidelberg (2014)
6. Applebaum, B.: Cryptography in Constant Parallel Time. Information Security and Cryptography. Springer, Heidelberg (2014)
7. Applebaum, B., Brakerski, Z.: Obfuscating circuits via composite-order graded encoding. In: Dodis, Y., Nielsen, J.B. (eds.) TCC 2015, Part II. LNCS, vol. 9015, pp. 528–556. Springer, Heidelberg (2015)
8. Applebaum, B., Ishai, Y., Kushilevitz, E.: Cryptography in NC^0 . In: FOCS, pp. 166–175 (2004)
9. Applebaum, B., Ishai, Y., Kushilevitz, E.: On pseudorandom generators with linear stretch in NC^0 . Comput. Complex. **17**(1), 38–69 (2008)
10. Applebaum, B., Lovett, S.: Algebraic attacks against random local functions and their countermeasures. In: Electronic Colloquium on Computational Complexity (ECCC), vol. 22, p. 172 (2015)
11. Banerjee, A., Peikert, C., Rosen, A.: Pseudorandom functions and lattices. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 719–737. Springer, Heidelberg (2012)
12. Barak, B., Garg, S., Kalai, Y.T., Paneth, O., Sahai, A.: Protecting obfuscation against algebraic attacks. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 221–238. Springer, Heidelberg (2014)
13. Barak, B., Goldreich, O., Impagliazzo, R., Rudich, S., Sahai, A., Vadhan, S.P., Yang, K.: On the (im)possibility of obfuscating programs. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 1–18. Springer, Heidelberg (2001)
14. Bitansky, N., Garg, S., Lin, H., Pass, R., Telang, S.: Succinct randomized encodings and their applications. In: Servedio, R.A., Rubinfeld, R. (eds.) 47th ACM STOC, Portland, OR, USA, June 14–17 2015, pp. 439–448. ACM Press (2015)
15. Bitansky, N., Paneth, O., Wichs, D.: Perfect structure on the edge of chaos - trapdoor permutations from indistinguishability obfuscation. In: Kushilevitz, E., et al. (eds.) TCC 2016-A. LNCS, vol. 9562, pp. 474–502. Springer, Heidelberg (2016). doi:[10.1007/978-3-662-49096-9_20](https://doi.org/10.1007/978-3-662-49096-9_20)
16. Bitansky, N., Vaikuntanathan, V.: Indistinguishability obfuscation from functional encryption. In: IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17–20 October 2015, pp. 171–190 (2015)
17. Bitansky, N., Vaikuntanathan, V.: Indistinguishability obfuscation: from approximate to exact. In: Kushilevitz, E., et al. (eds.) TCC 2016-A. LNCS, vol. 9562, pp. 67–95. Springer, Heidelberg (2016). doi:[10.1007/978-3-662-49096-9_4](https://doi.org/10.1007/978-3-662-49096-9_4)
18. Bogdanov, A., Qiao, Y.: On the security of goldreich’s one-way function. Comput. Complex. **21**(1), 83–127 (2012)
19. Boneh, D., Waters, B.: Constrained pseudorandom functions and their applications. In: Sako, K., Sarkar, P. (eds.) ASIACRYPT 2013, Part II. LNCS, vol. 8270, pp. 280–300. Springer, Heidelberg (2013)
20. Boneh, D., Wu, D.J., Zimmerman, J.: Immunizing multilinear maps against zeroizing attacks. Cryptology ePrint Archive, Report 2014/930 (2014). <http://eprint.iacr.org/2014/930>
21. Boyle, E., Goldwasser, S., Ivan, I.: Functional signatures and pseudorandom functions. In: PKC, pp. 501–519 (2014)
22. Brakerski, Z., Rothblum, G.N.: Virtual black-box obfuscation for all circuits via generic graded encoding. In: Lindell, Y. (ed.) TCC 2014. LNCS, vol. 8349, pp. 1–25. Springer, Heidelberg (2014)

23. Brakerski, Z., Vaikuntanathan, V.: Constrained key-homomorphic PRFs from standard lattice assumptions - or: how to secretly embed a circuit in your PRF. In: Dodis, Y., Nielsen, J.B. (eds.) TCC 2015, Part II. LNCS, vol. 9015, pp. 1–30. Springer, Heidelberg (2015)
24. Canetti, R., Goldwasser, S., Poburinnaya, O.: Adaptively secure two-party computation from indistinguishability obfuscation. In: Dodis, Y., Nielsen, J.B. (eds.) TCC 2015, Part II. LNCS, vol. 9015, pp. 557–585. Springer, Heidelberg (2015)
25. Canetti, R., Holmgren, J., Jain, A., Vaikuntanathan, V.: Succinct garbling and indistinguishability obfuscation for RAM programs. In: Servedio, R.A., Rubinfeld, R. (eds.) 47th ACM STOC, Portland, OR, USA, June 14–17 2015, pp. 429–437. ACM Press (2015)
26. Canetti, R., Lin, H., Tessaro, S., Vaikuntanathan, V.: Obfuscation of probabilistic circuits and applications. In: Dodis, Y., Nielsen, J.B. (eds.) TCC 2015, Part II. LNCS, vol. 9015, pp. 468–497. Springer, Heidelberg (2015)
27. Cheon, J.H., Han, K., Lee, C., Ryu, H., Stehlé, D.: Cryptanalysis of the multilinear map over the integers. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9056, pp. 3–12. Springer, Heidelberg (2015)
28. Chung, K., Lin, H., Pass, R.: Constant-round concurrent zero-knowledge from indistinguishability obfuscation. In: Gennaro, R., Robshaw, M. (eds.) CRYPTO 2015, Part I. LNCS, vol. 9215, pp. 287–307. Springer, Berlin (2015)
29. Cook, J., Etesami, O., Miller, R., Trevisan, L.: Goldreich’s one-way function candidate and myopic backtracking algorithms. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 521–538. Springer, Heidelberg (2009)
30. Coron, J.-S., et al.: Zeroizing without low-level zeroes: new MMAP attacks and their limitations. In: Gennaro, R., Robshaw, M.J.B. (eds.) CRYPTO 2015, Part I. LNCS, vol. 9215, pp. 247–266. Springer, Heidelberg (2015)
31. Coron, J.-S., Lepoint, T., Tibouchi, M.: Practical multilinear maps over the integers. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part I. LNCS, vol. 8042, pp. 476–493. Springer, Heidelberg (2013)
32. Coron, J.-S., Lepoint, T., Tibouchi, M.: New multilinear maps over the integers. In: Gennaro, R., Robshaw, M.J.B. (eds.) CRYPTO 2015, Part I. LNCS, vol. 9215, pp. 267–286. Springer, Heidelberg (2015)
33. Cryan, M., Miltersen, P.B.: On pseudorandom generators in NC0. In: Sgall, J., Pultr, A., Kolman, P. (eds.) MFCS 2001. LNCS, vol. 2136, p. 272. Springer, Heidelberg (2001)
34. Dachman-Soled, D., Katz, J., Rao, V.: Adaptively secure, universally composable, multiparty computation in constant rounds. In: Dodis, Y., Nielsen, J.B. (eds.) TCC 2015, Part II. LNCS, vol. 9015, pp. 586–613. Springer, Heidelberg (2015)
35. Garg, S., Gentry, C., Halevi, S.: Candidate multilinear maps from ideal lattices. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 1–17. Springer, Heidelberg (2013)
36. Garg, S., Gentry, C., Halevi, S., Raykova, M., Sahai, A., Waters, B.: Candidate indistinguishability obfuscation and functional encryption for all circuits. In: 54th FOCS, Berkeley, CA, 26–29 October 2013, pp. 40–49. IEEE Computer Society Press (2013)
37. Garg, S., Polychroniadou, A.: Two-round adaptively secure MPC from indistinguishability obfuscation. In: Dodis, Y., Nielsen, J.B. (eds.) TCC 2015, Part II. LNCS, vol. 9015, pp. 614–637. Springer, Heidelberg (2015)
38. Gentry, C., Gorbunov, S., Halevi, S.: Graph-induced multilinear maps from lattices. In: Dodis, Y., Nielsen, J.B. (eds.) TCC 2015, Part II. LNCS, vol. 9015, pp. 498–527. Springer, Heidelberg (2015)

39. Gentry, C., Halevi, S., Maji, H.K., Sahai, A.: Zeroizing without zeroes: cryptanalyzing multilinear maps without encodings of zero. Cryptology ePrint Archive, Report 2014/929 (2014). <http://eprint.iacr.org/2014/929>
40. Gentry, C., Lewko, A., Sahai, A., Waters, B.: Indistinguishability obfuscation from the multilinear subgroup elimination assumption. Cryptology ePrint Archive, Report 2014/309 (2014). <http://eprint.iacr.org/2014/309>
41. Goldreich, O.: Candidate one-way functions based on expander graphs. Cryptology ePrint Archive, Report 2000/063 (2000). <http://eprint.iacr.org/2000/063>
42. Goldreich, O., Goldwasser, S., Micali, S.: How to construct random functions. J. ACM **33**(4), 792–807 (1986)
43. Goldwasser, S., Kalai, Y.T., Popa, R.A., Vaikuntanathan, V., Zeldovich, N.: Reusable garbled circuits and succinct functional encryption. In: Boneh, D., Roughgarden, T., Feigenbaum, J. (eds.) 45th ACM STOC, Palo Alto, CA, USA, 1–4 June 2013, pp. 555–564. ACM Press (2013)
44. Ishai, Y., Kushilevitz, E.: Perfect constant-round secure computation via perfect randomizing polynomials. In: Widmayer, P., Triguero, F., Morales, R., Hennessy, M., Eidenbenz, S., Conejo, R. (eds.) ICALP 2002. LNCS, vol. 2380, pp. 244–256. Springer, Heidelberg (2002)
45. Kiayias, A., Papadopoulos, S., Triandopoulos, N., Zacharias, T.: Delegatable pseudorandom functions and applications. In: CCS, pp. 669–684 (2013)
46. Komargodski, I., Moran, T., Naor, M., Pass, R., Rosen, A., Yogev, E.: One-way functions and (im)perfect obfuscation (2014)
47. Koppula, V., Lewko, A.B., Waters, B.: Indistinguishability obfuscation for turing machines with unbounded memory. In: Servedio, R.A., Rubinfeld, R. (eds.) 47th ACM STOC, Portland, OR, USA, 14–17 June 2015, pp. 419–428. ACM Press (2015)
48. Lin, H., Pass, R., Seth, K., Telang, S.: Output-compressing randomized encodings and applications. IACR Cryptology ePrint Archive 2015:720 (2015)
49. Lin, H., Pass, R., Seth, K., Telang, S.: Indistinguishability obfuscation with non-trivial efficiency. In: Proceedings of the Public-Key Cryptography - PKC 2016 - 19th IACR International Conference on Practice and Theory in Public-Key Cryptography, Taipei, Taiwan, 6–9 March 2016, Part II, pp. 447–462 (2016)
50. Linial, N., Mansour, Y., Nisan, N.: Constant depth circuits, fourier transform, and learnability. In: 30th FOCS, pp. 574–579 (1989)
51. Mahmoody, M., Mohammed, A., Nematihaji, S.: More on impossibility of virtual black-box obfuscation in idealized models. Cryptology ePrint Archive, Report 2015/632 (2015). <http://eprint.iacr.org/2015/632>
52. Mossel, E., Shpilka, A., Trevisan, L.: On e-biased generators in NC0. In: Proceedings of the 44th Symposium on Foundations of Computer Science (FOCS 2003), Cambridge, MA, USA, 11–14 October 2003, pp. 136–145 (2003)
53. Naor, M., Reingold, O.: Synthesizers and their application to the parallel construction of pseudo-random functions. In: 36th FOCS, Milwaukee, Wisconsin, 23–25 October 1995, pp. 170–181. IEEE Computer Society Press (1995)
54. Naor, M., Reingold, O.: Number-theoretic constructions of efficient pseudo-random functions. In: 38th FOCS, Miami Beach, Florida, 19–22 October 1997, pp. 458–467. IEEE Computer Society Press (1997)
55. Naor, M., Reingold, O., Rosen, A.: Pseudo-random functions and factoring (extended abstract). In: 32nd ACM STOC, Portland, Oregon, USA, 21–23 May 2000, pp. 11–20. ACM Press (2000)
56. Nisan, N., Szegedy, M.: On the degree of boolean functions as real polynomials. Comput. Complex. **4**, 301–313 (1994)

57. O'Donnell, R., Witmer, D.: Goldreich's PRG: evidence for near-optimal polynomial stretch. In: IEEE 29th Conference on Computational Complexity, CCC 2014, Vancouver, BC, Canada, 11–13 June 2014, pp. 1–12 (2014)
58. Pass, R., 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>
59. Pass, R., Seth, K., Telang, S.: Indistinguishability obfuscation from semantically-secure multilinear encodings. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014, Part I. LNCS, vol. 8616, pp. 500–517. Springer, Heidelberg (2014)
60. Sahai, A., Waters, B.: How to use indistinguishability obfuscation: deniable encryption, and more. In: Shmoys, D.B. (ed.) 46th ACM STOC, New York, NY, USA, 31 May – 3 June 2014, pp. 475–484. ACM Press (2014)
61. Sahai, A., Waters, B.: How to use indistinguishability obfuscation: deniable encryption, and more. In: Proceedings of the STOC 2014 (2014)
62. Zimmerman, J.: How to obfuscate programs directly. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9057, pp. 439–467. Springer, Heidelberg (2015)

Advances in Cryptology – EUROCRYPT 2016
35th Annual International Conference on the Theory
and Applications of Cryptographic Techniques, Vienna,
Austria, May 8–12, 2016, Proceedings, Part I
Fischlin, M.; Coron, J.-S. (Eds.)
2016, XXVIII, 853 p. 155 illus., Softcover
ISBN: 978-3-662-49889-7