INDISTINGUISHABILITY OBFUSCATION

AAYUSH JAIN, HUIJIA LIN, AND AMIT SAHAI

ABSTRACT

At least since the initial public proposal of public-key cryptography based on computational hardness conjectures (Diffie and Hellman, 1976), cryptographers have contemplated the possibility of a "one-way compiler" that translates computer programs into "incomprehensible" but equivalent forms. And yet, the search for such a "one-way compiler" remained elusive for decades. We examine a formalization of this concept with the notion of indistinguishability obfuscation ($i\mathcal{O}$). Roughly speaking, $i\mathcal{O}$ requires that the compiled versions of any two equivalent programs (with the same size and running time) be indistinguishable to any efficient adversary. Finally, we show how to construct $i\mathcal{O}$ in such a way that we can prove the security of our $i\mathcal{O}$ scheme based on well-studied computational hardness conjectures in cryptography.

MATHEMATICS SUBJECT CLASSIFICATION 2020

Primary 94A60; Secondary 68Q17, 68Q17, 11H99, 11T71

KEYWORDS

Indistinguishability obfuscation, well-studied assumptions, learning parity with noise, pairing groups, pseudorandom generators, learning with errors

INTERNATIONAL CONGRESS OF MATHEMATICIANS
2022 JULY 6–14

Proc. Int. Cong. Math. 2022, Vol. 6, pp. 4762–4798
DOI 10.4171/ICM2022/86

Published by EMS Press

1. INTRODUCTION

Consider the polynomial $f_1(x,y) \in \mathbb{Z}[x,y]$ that is computed as follows:

$$f_1(x,y) = (x+y)^{16} - (x-y)^{16}.$$

Alternatively, contemplate the polynomial $f_2(x,y) \in \mathbb{Z}[x,y]$ that is computed via:

$$\begin{aligned} f_2(x,y) = 32x^{15}y + 1120x^{13}y^3 + 8736x^{11}y^5 + 22880x^9y^7 \\ + 22880x^7y^9 + 8736x^5y^{11} + 1120x^3y^{13} + 32xy^{15}. \end{aligned}$$

A calculation shows that f_1 and f_2 are, in fact, the same polynomial, computed in two different ways. Indeed, the expressions f_1 and f_2 above are special cases of *arithmetic circuits*, which precisely represent "ways to compute a polynomial."

What if we wanted to hide all implementation choices made when creating such an arithmetic circuit for a particular polynomial? An easy way to do that would be to first convert our polynomial into a canonical form, and then implement the canonical form as an arithmetic circuit. Indeed, the description of f_2 above can be seen as a canonical representation of the polynomial as a sum of monomials with regard to a natural monomial ordering. However, as this example illustrates, canonical forms can be substantially more complex than other implementations of the same polynomial. For polynomials in n variables, the loss in efficiency can be exponential in n. This would often make computing the canonical form—or indeed, even writing it down—infeasible.

A pseudocanonical form. Given that computing canonical forms can be infeasible, what is there to do? Here, following [22], we draw an analogy to the notion of pseudorandomness. When truly random values are not available, we can instead aim to produce values that "look random" by means of a pseudorandom generator. That is, we require that no efficient algorithm can distinguish between truly random values and the output of our pseudorandom generator.

Now, for two arithmetic circuits g_1 and g_2 that compute the same underlying polynomial, a true canonical form $\mathsf{Canonical}(g_1)$ would be identical to the canonical form of $\mathsf{Canonical}(g_2)$. Instead, we would ask that a pseudocanonical form $\mathsf{PseudoCanonical}(g_1)$ would simply be indistinguishable from the pseudocanonical form $\mathsf{PseudoCanonical}(g_2)$, to all efficient algorithms that were given g_1 and g_2 as well. Observe that unless there are actual efficiently computable canonical forms for all arithmetic circuits—which we do not believe to be true—it must be that such a $\mathsf{PseudoCanonical}$ operator is randomized, and outputs a *probability distribution* over arithmetic circuits computing the same polynomial.

The computing lens. Let us now step back, and view the problem stated above through the lens of computing. The classic theory of computation (see, e.g., [46]) tells us that general computer programs can be converted into equivalent polynomials (albeit over finite fields, which we will focus on implicitly in the sequel). So the pseudocanonicalization question posed above is equivalent to the pseudocanonicalization question for general computer programs. Indeed, the question of hiding implementation details within a computer program has a long history, dating at least as far back as the groundbreaking 1976 work of [50] introducing the concept of public-key cryptography. Historically, this problem has been called "program obfuscation," albeit it was typically discussed in an ill-defined form. Discussed in these vague terms, it was folklore that truly secure program obfuscation would have revolutionary applications to computing, especially for securing intellectual property. The work of [22] gave a formal treatment of this problem, and proved the impossibility of strong forms of general-purpose program obfuscation. This work also formalized the pseudo-canonicalization problem discussed above via the notion of *indistinguishability obfuscation* ($i\mathcal{O}$). Writing now in the language of Boolean circuits, we define the problem as follows:

Definition 1.1 (Indistinguishability obfuscator (iO) for circuits [22]). A probabilistic polynomial-time algorithm $i\mathcal{O}$ is called a secure indistinguishability obfuscator for polynomial-sized circuits if the following holds:

- (*Completeness*) For every $\lambda \in \mathbb{N}$, every circuit C with input length n, and every input $x \in \{0,1\}^n$, we have that
$$\Pr\big[\tilde{C}(x) = C(x) : \tilde{C} \leftarrow i\mathcal{O}(1^\lambda, C)\big] = 1.$$
- (*Indistinguishability*) For every two ensembles $\{C_{0,\lambda}\}_{\lambda\in\mathbb{Z}^+}$ and $\{C_{1,\lambda}\}_{\lambda\in\mathbb{Z}^+}$ of polynomial-sized circuits that have the same size, input length, and output length, and are functionally equivalent, that is, $\forall\lambda \in \mathbb{Z}^+$, $C_{0,\lambda}(x) = C_{1,\lambda}(x)$ for every input x, the distributions $i\mathcal{O}(1^\lambda, C_{0,\lambda})$ and $i\mathcal{O}(1^\lambda, C_{1,\lambda})$ are computationally indistinguishable, that is, for every efficient polynomial-time algorithm D and for every constant $c > 0$, there exists a constant $\lambda_0 \in \mathbb{Z}^+$ such that, for all $\lambda > \lambda_0$, we have
$$\big|\Pr\big[D(i\mathcal{O}(1^\lambda, C_{0,\lambda}) = 1\big] - \Pr\big[D(i\mathcal{O}(1^\lambda, C_{1,\lambda}) = 1\big]\big| \le \frac{1}{\lambda^c}.$$

As we discuss below in Section 1.2, indeed $i\mathcal{O}$ as a formalization of pseudo-canonicalization lived up to the folklore promise of software obfuscation: there was, and still is, a large research community studying novel applications of $i\mathcal{O}$.

In contrast, demonstrating the feasibility of constructing $i\mathcal{O}$ proved far more challenging. Often one expects that theory will lag behind practice, and given the folklore promise of software obfuscation, one might expect that over the years perhaps clever programmers had come up with heuristic approaches to software obfuscation that resisted attack. The reality is the opposite. Indeed, in 2021 the third annual White Box Cryptography contest was held to evaluate heuristic methods for software obfuscation, and every one of the 97 submitted obfuscations was broken before the contest ended [44].

A large body of theoretical work, starting with the pioneering work of [55], has attempted to construct $i\mathcal{O}$ using mathematical tools. However, prior to the result [68] by the authors of this article, all previous mathematical approaches to constructing $i\mathcal{O}$ relied on new, unproven mathematical assumptions, many of which turned out to be false. We survey this work in Section 1.3 below.

We would like to build $i\mathcal{O}$ whose security rests upon cryptographic hardness assumptions that have stood the test of the time, have a long history of study, and are widely believed to be true. The main result of our works [68, 69] is the construction of an $i\mathcal{O}$ scheme from three well-studied assumptions. We discuss this in more detail next.

Informal Theorem 1.1 ([68, 69]). Under the following assumptions[1]:

- *the Learning Parity with Noise (*LPN*) assumption over general prime fields $\mathbb{Z}_p$ with polynomially many* LPN *samples and error rate $1/k^\delta$, where k is the dimension of the* LPN *secret, and $\delta > 0$ is any constant;*
- *the existence of a Boolean Pseudorandom Generator (*PRG*) in* NC^0 *with stretch $n^{1+\tau}$, where n is the length of the* PRG *seed, and $\tau > 0$ is any constant;*
- *the Decision Linear (*DLIN*) assumption on symmetric bilinear groups of prime order,*

indistinguishability obfuscation for all polynomial-size circuits exists.

The three assumptions above (discussed further below in Section 1.1) are based on computational problems with a long history of study, rooted in complexity, coding, and number theory. Further, they were introduced for building basic cryptographic primitives (such as public key encryption), and have been used for realizing a variety of cryptographic goals that have nothing to do with $i\mathcal{O}$.

1.1. Assumptions in more detail

We now describe each of the assumptions we need in more detail and briefly survey their history.

The DLIN assumption. The Decisional Linear assumption (DLIN) is stated as follows: For an appropriate λ-bit prime p, two groups $\mathbb{G}$ and $\mathbb{G}_T$ of order p are chosen such that there exists an efficiently computable nontrivial symmetric bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. A canonical generator g for $\mathbb{G}$ is also computed. Following the tradition of cryptography, we describe the groups above using multiplicative notation, even though they are cyclic. The DLIN assumption requires that the following computational indistinguishability holds:

$$\begin{aligned} &\{(g^x, g^y, g^{xr}, g^{ys}, g^{r+s}) \mid x, y, r, s \leftarrow \mathbb{Z}_p\} \\ &\quad \approx_c \{(g^x, g^y, g^{xr}, g^{ys}, g^z) \mid x, y, r, s, z \leftarrow \mathbb{Z}_p\}. \end{aligned}$$

This assumption was first introduced in the 2004 work of Boneh, Boyen, and Shacham [31], and instantiated using appropriate elliptic curves. Since then DLIN and assumptions implied by DLIN have seen extensive use in a wide variety of applications throughout cryptography, such as Identity-Based Encryption, Attribute-Based Encryption, Functional Encryption for degree 2 polynomials, Noninteractive Zero Knowledge, etc. (see, e.g., [25, 38, 62, 89]).

---

1 For technical reasons, we need to hardness of these assumptions to be such that no polynomial-time adversaries have beyond subexponentially small advantage in breaking the hardness of the underlying problems.

The existence of PRGs in NC^0. The assumption of the existence of a Boolean Pseudorandom Generator PRG in NC^0 states that there exists a Boolean function $\mathsf{G} : \{0,1\}^n \to \{0,1\}^m$ where $m = n^{1+\tau}$ for some constant $\tau > 0$, and where each output bit computed by G depends on a constant number of input bits, such that the following computational indistinguishability holds:

$$\{\mathsf{G}(\boldsymbol{\sigma}) \mid \boldsymbol{\sigma} \leftarrow \{0,1\}^n\} \approx_c \{\boldsymbol{y} \mid \boldsymbol{y} \leftarrow \{0,1\}^m\}.$$

Pseudorandom generators are a fundamental primitive in their own right, and have vast applications throughout cryptography. PRGs in NC^0 are tightly connected to the fundamental topic of Constraint Satisfaction Problems (CSPs) in complexity theory, and were first proposed for cryptographic use by Goldreich [49,61,65] 20 years ago. The complexity theory and cryptography communities have jointly developed a rich body of literature on the cryptanalysis and theory of constant-locality Boolean PRGs [10,12,13,16,17,30,45,48,49,61,73,86,87].

LPN over large fields. The Learning Parity with Noise LPN assumption over finite fields $\mathbb{Z}_p$ is a decoding problem. The standard LPN assumption with respect to subexponential-size modulus p, dimension ℓ, sample complexity n, and a noise rate $r = 1/\ell^\delta$ for some $\delta \in (0,1)$ states that the following computational indistinguishability holds:

$$\begin{aligned}&\{\boldsymbol{A}, \boldsymbol{s}\cdot\boldsymbol{A} + \boldsymbol{e} \bmod p \mid \boldsymbol{A} \leftarrow \mathbb{Z}_p^{\ell\times n}, \boldsymbol{s} \leftarrow \mathbb{Z}_p^{1\times\ell}, \boldsymbol{e} \leftarrow \mathcal{D}_r^{1\times n}\}\\ &\quad\approx_c \{\boldsymbol{A}, \boldsymbol{u} \mid \boldsymbol{A} \leftarrow \mathbb{Z}_p^{\ell\times n}, \boldsymbol{u} \leftarrow \mathbb{Z}_p^{1\times n}\}.\end{aligned}$$

Above $e \leftarrow \mathcal{D}_r$ is a generalized Bernoulli distribution, i.e., e is sampled randomly from $\mathbb{Z}_p$ with probability $1/\ell^\delta$ and set to be 0 with probability $1 - 1/\ell^\delta$. We consider polynomial sample complexity $n(\ell)$, and the modulus p is an arbitrary subexponential function in ℓ.

The origins of the LPN assumption date all the way back to the 1950s: the works of Gilbert [60] and Varshamov [95] showed that random linear codes possessed remarkably strong minimum distance properties. However, since then, very little progress has been made in efficiently decoding random linear codes under random errors. The LPN over fields assumption above formalizes this, and was introduced over $\mathbb{Z}_2$ for cryptographic uses in 1994 [29], and formally defined for general finite fields and parameters in 2009 [66], under the name "Assumption 2."

While in [66] the assumption was used when the error rate was assumed to be a constant, in fact, polynomially low error (in fact, $\delta = 1/2$) has an even longer history in the LPN literature: it was used by Alekhnovitch in 2003 [4] to construct public-key encryption with the field $\mathbb{Z}_2$, and used to build public-key encryption over $\mathbb{Z}_p$ in 2015 [11]. The exact parameter settings that we describe above, with both general fields and inverse polynomial error rate corresponding to an arbitrarily small constant $\delta > 0$, were explicitly posed by [35], in the context of building efficient secure two-party and multiparty protocols for arithmetic computations.

Recently, the LPN assumption has led to a wide variety of applications (see, for example, [11, 14, 35, 37, 52, 59, 66]). A comprehensive review of known attacks on LPN over large fields, for the parameter settings we are interested in, was given in [35, 36]. For our parameter setting, the running time of all known attacks is $\Omega(2^{\ell^{1-\delta}})$, for any choice of the constant $\delta \in (0,1)$ and for any polynomial number of samples $n(\ell)$.

On search vs. decision versions of our assumptions. Except for the DLIN assumption, the other two assumptions that we make can be based on search assumptions.

The LPN over $\mathbb{Z}_p$ assumption we require is implied by the subexponential hardness of its corresponding search versions [29,82,83,91]. As summarized in [94], there is a search-to-decision reduction[2] whose sample complexity is $m = \mathsf{poly}(\dim(\boldsymbol{s}), m', 1/\varepsilon)$ (namely, polynomial in the dimension $\dim(\boldsymbol{s})$ of the secret, sample complexity m' of the decision version, and the inverse of the distinguishing gap ε) and runtime $\mathsf{poly}(\dim(\boldsymbol{s}), p, m)$. In this work, we need the pseudorandomness of (polynomially many) LPN samples to hold against polynomial-time adversaries, with a *subexponential* distinguishing gap. We can further set the modulus p to an arbitrarily small subexponential function[3] in $\dim(\boldsymbol{s})$. Decisional LPN with such parameters are implied by the following subexponential search LPN assumption: There is a constant $\gamma > 0$ such that no subexponential-time $2^{\dim(\boldsymbol{s})^\gamma}$ adversary, given a subexponential $2^{\dim(\boldsymbol{s})^\gamma}$ number of samples, can recover $\boldsymbol{s}$ with noticeable probability.

The works of [10,16] showed that the one-wayness of *random local functions* implies the existence of PRGs in NC^0. More precisely, for a length parameter $m = m(n)$, a locality parameter $d = O(1)$, and a d-ary predicate $Q : \{0,1\}^d \to \{0,1\}$, a distribution $\mathcal{F}_{Q,m}$ samples a d-local function $f_{G,Q} : \{0,1\}^d \to \{0,1\}$ by choosing a random d-uniform hypergraph G with n nodes and m hyperedges, where each hyperedge is chosen uniformly and independently at random. The ith output bit of $f_{G,Q}$ is computed by evaluating Q on the d input bits indexed by nodes in the ith hyperedge. The one-wayness of $\mathcal{F}_{Q,m}$ for proper choices of Q, m has been conjectured and studied in [12,45,61,86]. The works of [10,16] showed how to construct a family of PRG in NC^0 with polynomial stretch based on the one-wayness of $\mathcal{F}_{Q,m}$ for any Q that is sensitive (i.e., some input bit i of Q has full influence) and any $m = n^{1+\delta}$ with $\delta > 0$. The constructed PRGs have negligible distinguishing advantage and the reduction incurs a multiplicative polynomial security loss. Therefore, the subexponential pseudorandomness of PRG in NC^0 that we need is implied by the existence of $\mathcal{F}_{Q,m}$ that is hard to invert with noticeable probability by adversaries of some subexponential size.

1.2. Applications of $i\mathcal{O}$

The notion of $i\mathcal{O}$ occupies an intriguing and influential position in complexity theory and cryptography. Interestingly, if $\mathsf{NP} \subseteq \mathsf{BPP}$, then $i\mathcal{O}$ exists for the class of all polynomial-size circuits because if $\mathsf{NP} \subseteq \mathsf{BPP}$, then it is possible to efficiently compute a canonical form for any function computable by a polynomial-size circuit. On the other hand, if $\mathsf{NP} \not\subseteq \mathsf{io\text{-}BPP}$, then in fact the existence of $i\mathcal{O}$ for polynomial-size circuits implies that one-way functions exist [71]. A large body of work has shown that $i\mathcal{O}$ plus one-way func-

2 Importantly, this reduction is oblivious to the distribution of the errors and hence applies to both LWE and LPN.

3 In the construction, we set $p = \Theta(2^\lambda)$ and $\dim(\boldsymbol{s})$ to a large enough polynomial in λ.

tions imply a vast array of cryptographic objects, so much so that iO has been conjectured to be a "central hub" [71,92] for cryptography.

An impressive list of fascinating new cryptographic objects are only known under $i\mathcal{O}$ or related objects such as functional encryption and witness encryption. Hence, our construction of $i\mathcal{O}$ from well-founded assumptions immediately implies these objects from the same assumptions. Below, we highlight a small subset of these implications as corollaries. In all the applications, by λ we denote the security parameter.

Corollary 1.1 (Informal). *Assuming the* subexponential *hardness of the three assumptions in Theorem 1.1, we have:*

- *Multiparty noninteractive key exchange in the plain model (without trusted setup), e.g., [33,70];*
- *Selectively sound and perfectly zero-knowledge Succinct Noninteractive ARGument (ZK-SNARG) for any NP language with statements up to a bounded polynomial size in the CRS model, where the CRS size is* $\mathsf{poly}(\lambda)(n+m)$*,* n, m *are upper bounds on the lengths of the statements and witnesses, and the proof size is* $\mathsf{poly}(\lambda)$ *[92];*
- *(Symmetric or asymmetric) multilinear maps with bounded polynomial multilinear degrees, following [3,53], and a self-bilinear map over composite and unknown order group, assuming additionally the polynomial hardness of factoring [97];*
- *Witness Encryption (WE) for any NP language, following as a special case of* $i\mathcal{O}$ *for polynomial size circuits;*
- *Secret sharing for any monotone function in NP [72];*
- *Fully homomorphic encryption scheme for unbounded-depth polynomial size circuits (without relying on circular security), assuming slightly superpolynomial hardness of the assumptions above [41];*
- *Hardness of finding Nash equilibrium (more generally, for the class* PPAD*) [27].*

1.3. Prior work on the feasibility of $i\mathcal{O}$

There is a rich landscape of research on conjectured constructions of $i\mathcal{O}$. Despite being posed as a question at least 20 years ago [22,50], the first candidate mathematical construction came only in 2013, through the work of [55]. This construction relied on a newly constructed primitive called multilinear maps [54], which is a generalization of a bilinear maps where one could do high degree computations in the exponents. Soon after, several different candidates for multilinear maps were proposed [47,57] and many other constructions of $i\mathcal{O}$ were proposed. This propelled a huge body of constructions of $i\mathcal{O}$ relying on multilinear maps and related ideas (e.g., [18,20,39,43,47,51,54,55,57,84,85,90].) Unfortunately, all these works suffered from one of the three main problems:

- Most constructions were heuristic in the sense that they were just conjectured to be secure. There was no simple assumption on the multilinear maps on which you could base security.

- Sometimes security was based on some new assumption, but it was a new assumption proposed solely for proving that the construction was secure. Such assumptions lacked a long history of study.

- Most of the time, in the above both cases there were actually cycles of attacks and fixes on the constructions and/or underlying assumptions (e.g., [19,21,42,43,63,81,84,85]) which reduced our confidence further.

With this, the focus shifted to trying to minimize the degree of the multilinear map needed, with the goal of eventually reaching degree 2. In a beautiful line of work [9,74,75,79,80], it was shown that $i\mathcal{O}$ can be constructed just from succinct assumptions on degree-3 multilinear maps. Unfortunately, the candidates for degree-3 multilinear maps were the same as the candidates for high-degree multilinear maps and suffered from the same class of attacks as before.

Soon after, a line of work [1,2,6,8,56,67,76] constructed $i\mathcal{O}$ relying on bilinear maps, along with new kinds of pseudorandom generators. These assumptions were much simpler to state than before. Even though earlier proposals for some of those pseudorandom generators were attacked [19,21,81], exploring the limits of those attacks helped us design $i\mathcal{O}$ based on new but simple-to-state assumptions [6,56,67] that resisted all known attacks. However, these assumptions were newly stated and did not have a long history of study.

Therefore, building upon [6,8,56,67,76], these works culminated finally in our recent works [68,69], which managed to construct $i\mathcal{O}$ from the three assumptions in Theorem 1.1. This eliminated the need for making any new unstudied hardness assumptions. We now discuss some of the main open problems in the space of $i\mathcal{O}$ constructions.

1.4. Open problems

Our work places $i\mathcal{O}$ on firm foundations with respect to the assumptions it is based on, thereby answering the main feasibility question for the primitive (until we resolve the P vs. NP question). However, there are many important open questions that remain to be answered:

- *Concrete efficiency.* Our work first builds the notion of functional encryption and then boosts this object to $i\mathcal{O}$ via a complex transformation [5,28]. As a result, the final construction is quite complex. A highly important question that remains open is the following one: Is it possible to construct $i\mathcal{O}$ either by fine-tuning our approach, or otherwise (as in [23,58]) in a way that the resulting scheme yields concrete implementable efficiency? For this question, as a first step, it is even interesting if the construction rests upon new assumptions as long as the assumptions are rigorously cryptanalyzed.

- *Postquantum $i\mathcal{O}$.* Our work relies on bilinear maps (in a somewhat crucial way). As a result of that, our construction is broken in polynomial time using a quantum computer. Therefore, an important and a natural question to ask here is if we can build $i\mathcal{O}$ on any combination of well-studied postquantum assumptions such as LWE, LPN, or PRG in NC^0. This is indeed an active area of research.
- *$i\mathcal{O}$ for quantum circuits.* All known constructions of $i\mathcal{O}$ support only classical circuits. If quantum computers come one day, an interesting question is to construct an $i\mathcal{O}$ scheme that can be used to actually obfuscate quantum circuits. There are some results in restricted models [24, 40], but none of the known constructions work to obfuscate general quantum programs.
- *Understanding assumptions better.* We are still in the early stages of understanding the feasibility of $i\mathcal{O}$. An immediate question that arises out of work is to identify essential and nonessential assumptions out of the three assumptions, and if any of the assumptions can be replaced by another. Identifying if there is any other substantially different approach that also yields $i\mathcal{O}$ from well-studied assumptions will also shed light on this question.

2. TECHNICAL OVERVIEW: HOW TO CONSTRUCT $i\mathcal{O}$?

Below, we describe a very high-level overview of the main technical ideas implying $i\mathcal{O}$. For simplicity of exposition, we choose the simplest path to $i\mathcal{O}$ that we are aware of. This overview is based on a combination of ideas from [68] and [69]. However, for simplicity, the route discussed below would require one additional assumption to the three stated above (See Theorem 1.1)—namely, the Learning With Errors (LWE) [91] assumption. However, we do not actually discuss the exact technical reasons for needing LWE, as this assumption is actually unnecessary [69].

2.1. Preliminaries

Let us start with introducing some basic notation. Let $\mathsf{size}(X)$ indicate the length of the binary description of an object X (e.g., a string, a circuit, or truth table). Throughout, we consider Boolean functions or circuits or algorithms mapping n-bit binary strings to m-bit binary strings, for some $n, m \in \mathbb{Z}^+$. Let $\mathsf{time}(A, x)$ denote the running time of an algorithm (or circuit) A on an input x (in the case of a circuit C, $\mathsf{time}(C, x)$ is the same as $\mathsf{size}(C)$). We say that an algorithm or circuit is efficient if its running time is bounded by a (fixed) polynomial in the length of the input, that is, $\mathsf{time}(C, x) = \mathsf{size}(x)^c$ for some positive integer $c \in \mathbb{Z}^+$. When we only care about the existence of a constant and the concrete value is not important, we write $O(1)$ in place of that constant, e.g., $\mathsf{time}(C, x) = \mathsf{size}(x)^{O(1)}$ (following the big-O notation in complexity theory).

Our goal is designing an efficient *randomized* algorithm, called the obfuscator $\mathcal{O}$, that, given a Boolean circuit $C : \{0,1\}^n \to \{0,1\}$ with size $s \le n^{O(1)}$, referred to as the original circuit, outputs another Boolean circuit $\hat{C} : \{0,1\}^n \to \{0,1\}$, called the obfuscated circuit, such that the following three properties hold:

- (*Correctness*) The obfuscated circuit $\hat{C}$ is *functionally equivalent* to the original circuit C, denoted as $\hat{C} \equiv C$, meaning that for every $x \in \{0,1\}^n$, $\hat{C}(x) = C(x)$. Correctness must hold no matter what random coins the obfuscator $\mathcal{O}$ uses.
- (*Efficiency*) The obfuscator is efficient, meaning $\mathcal{O}$ runs in time polynomial in the size of the original circuit, namely, $\mathsf{time}(\mathcal{O}, C) = \mathsf{size}(C)^{O(1)}$.
- (*Security*) The obfuscated circuit $\hat{C}$ hides the implementation details in the original circuit C. This is formalized as follows: for every two *equally-sized* and *functionally-equivalent* circuits C_0 and C_1 (i.e., $\mathsf{size}(C_0) = \mathsf{size}(C_1)$ and $C_0 \equiv C_1$), the obfuscated circuits $\hat{C}_0$ and $\hat{C}_1$ are *computationally hard to distinguish*.

In the above by *distinguish* we mean having an algorithm D acting as a distinguisher and which, given an obfuscated circuit $\hat{C}$ generated from C_0 or C_1 chosen at random with equal probability, tries to determine which of C_0 and C_1 is the original circuit. By *computationally hard* we mean that no *efficient* distinguisher D can do much better than random guessing, that is, the probability of guessing correctly is bounded by $\frac{1}{2} + \varepsilon$ for some very small ε. And we say that $\hat{C}_0$ and $\hat{C}_1$ are computationally indistinguishable, which intuitively implies that they hide all implementation differences between C_0 and C_1 to computationally limited adversaries. However, computationally *unlimited* adversaries may well be able to distinguish them. We focus on computational security, since if $i\mathcal{O}$ with security against computationally unlimited adversaries were to exist, this would imply a collapse of the polynomial hierarchy [34] in complexity theory, a collapse which is widely conjectured to be false.

Next, we give an informal overview of how to construct $i\mathcal{O}$ from well-studied assumptions. In Section 2.2, we describe first how to reduce the task of constructing $i\mathcal{O}$ that compiles general Boolean circuits to a much simpler task—building $xi\mathcal{O}$ (introduced shortly) for specific simple circuits. In Section 2.3, we illustrate how this simplified task connects with *bilinear pairing groups*. This overview paints the overall blueprint. In the next section, Section 3.2, we will zoom into the key ideas that bridge the simpler task with bilinear pairing groups. These ideas are the last jigsaw pieces that complete the construction of $i\mathcal{O}$, which appeared in our latest works [68, 69].

2.2. Simplifying the task of $i\mathcal{O}$

Perhaps the simplest starting point is the following: If there is no restriction on the time the obfuscator $\mathcal{O}$ can take, then there is an extremely intuitive obfuscator: the obfuscated circuit is the *truth table* of the original circuit. The truth table TT_C of a Boolean circuit C is an array indexed by inputs, where $\mathsf{TT}_C[x] = C(x)$. It can be computed in time $2^n \cdot s$ if the input length is n and circuit size is s. Perfect security comes from the fact that, for any two functionally equivalent circuits $C_0 \equiv C_1$, their truth tables are identical $\mathsf{TT}_{C_0} = \mathsf{TT}_{C_1}$ and hence impossible to distinguish.

To put simply, a truth table is a *canonical form* of all circuits producing it. While outputting the truth table satisfies the correctness and security requirement of $i\mathcal{O}$, the obvious flaw with this is that the obfuscator is far from efficient: The running time of an $i\mathcal{O}$ scheme should be $s^{O(1)}$, rather than $2^n \cdot s$. The input length n of a Boolean circuit can be close to its size s, and hence the time to compute a truth table is exponentially large! This inefficiency is likely inherent, as efficient methods of finding canonical forms of circuits implies the collapse of the polynomial hierarchy, which is widely conjectured to be false.

Therefore, to improve efficiency, we seek canonical forms that fool computationally limited adversaries. Naturally, we start with a more humble goal:

Can we improve efficiency slightly to, say $2^{n(1-\varepsilon)} \cdot s^{O(1)}$ for some small $\varepsilon > 0$? What does $i\mathcal{O}$ with such nontrivial efficiency imply?

Simplification 1: obfuscation with nontrivial exponential efficiency. $i\mathcal{O}$ with nontrivial efficiency was studied in [26, 77, 78]. Surprisingly, their authors showed that very modest improvement on efficiency—captured in the notion of exponential-efficiency $i\mathcal{O}$, or $xi\mathcal{O}$ for short—is actually enough to construct completely efficient (polynomial time) $i\mathcal{O}$:

- $xi\mathcal{O}$ is an obfuscator $\mathcal{O}$ whose running time is still "trivial" $(2^n \cdot s)^{O(1)}$, but outputs an obfuscated circuit $\hat{C}$ of "nontrivial" size $2^{n(1-\varepsilon)} \cdot s^{O(1)}$ for some $\varepsilon > 0$.[4]

We can think of $xi\mathcal{O}$ (as well as $i\mathcal{O}$) as a special kind of encryption method, where the obfuscated circuit $\hat{C}$ is a "ciphertext" of the original circuit C, also denoted as $\mathsf{spCT}(C) = \hat{C}$, such that

- the ciphertext $\mathsf{spCT}(C)$ hides all information about C, except that it lets anyone with access to it learn the truth table of C. This is unlike normal encryption that reveals no information of the encrypted message.
- The size of the ciphertext $\mathsf{spCT}(C)$ is $2^{(1-\varepsilon)\cdot n}$ for some $0 < \varepsilon < 1$. So it can be viewed as a (slightly) compressed version of the truth table (that reveals no other information of C to computationally limited adversaries).

Such a special encryption scheme is known as *functional encryption* [32, 88, 93], which controls precisely which information of the encrypted message is revealed, and hides all other information. This notion is tightly connected with $xi\mathcal{O}$ and $i\mathcal{O}$, and, in fact, the implication of $xi\mathcal{O}$ to $i\mathcal{O}$ goes via the notion of functional encryption [5, 7, 28].

When viewing $\mathsf{spCT}(C)$ as a compressed version of the truth table. It becomes clear why even slight compression is powerful enough to imply $i\mathcal{O}$: The idea is keeping compressing iteratively until the size of the special ciphertext becomes polynomial. The works

4 We note that $xi\mathcal{O}$ should be distinguished from the Minimal Circuit Size Problem (MCSP) in complexity theory, which asks to compute the circuit complexity of a function, given as a truth table TT. In contrast, the obfuscator is given a small circuit C as input.

of [5,28,77,78] turn this high-level idea into an actual proof that $xi\mathcal{O}$ implies $i\mathcal{O}$[5] and allows us to focus on constructing $xi\mathcal{O}$, or equivalently, the special encryption described above.

Simplification 2: it suffices to obfuscate simple circuits. Unfortunately, despite the efficiency relaxation, it is still unclear how to obfuscate general Boolean circuits, which can be complex. Naturally, we ask:

> *Can we obfuscate simple subclasses of circuits?*
> *What does $xi\mathcal{O}$ for simple circuits imply?*

It turns out that it suffices to focus on an extremely simple class of circuits $\mathsf{C} = \mathsf{NC}^0$, where NC^0 is the set of all circuits with constant *output locality*, meaning every output bit depends on a constant number of input bits. To do so, we will rely on two cryptographic tools, *randomized encodings* and *Pseudorandom Generators (PRGs)*.

Randomized encoding in NC^0. A randomized encoding (RE) scheme consists of two efficient algorithms (RE, Decode). It gives a way to represent a complex circuit $C(\cdot)$ by a much simpler *randomized* circuit $\mathsf{RE}_C(\cdot;\cdot) := \mathsf{RE}(C,\cdot;\cdot)$ such that

- (*Correctness*) For every input x, the output π_x of $\mathsf{RE}(C, x; r)$ produced using uniformly random coins r encodes the correct output; in other words, there exists an efficient decoding algorithm Decode such that $\mathsf{Decode}(\pi) = C(x)$.
- (*Security*) π_x reveals no information of C beyond the output of x to efficient adversaries.
- (*Simplicity*) RE is a simple circuit by some measure of simplicity. Classic works [15,64,98] showed that RE can be simply an NC^0 circuit, when assuming the existence of PRGs in NC^0 like we are.

The correctness and security of the randomized encoding suggests that, instead of directly obfuscating a general circuit C, we can alternatively obfuscate a circuit D that on input x outputs an encoding π_x, which reveals only $C(x)$. The potential benefit is that D depending on RE and C should be simply an NC^0 circuit. Hence, it would suffice to construct $xi\mathcal{O}$ for simple NC^0 circuits!

To make the above idea go through, there are, however, a few wrinkles to be ironed out. The issue is that the security of randomized encoding only holds if an encoding π_x is generated using fresh random coins. There are concrete attacks that learn information of C beyond the outputs, when π_x is generated using nonuniform random coins, or when two encoding π_x and $\pi_{x'}$ are generated using correlated random coins. If the truth table of the

5 This implication, however, comes with a quantitative weakening in the security. To obtain $i\mathcal{O}$ that is secure against adversaries that run in time polynomial in the input length, the original $xi\mathcal{O}$ needs to be secure against adversaries that run in time subexponential $2^{n^{\varepsilon}}$ for some $\varepsilon \in (0,1)$ in its input length n.

circuit D contains an encoding π_x for every input x (i.e., $\mathsf{TT}_D[x] = \pi_x$) the random coins for generating these encoding must be embedded in D, that is,

$$D = D_{C,\mathsf{RE},r_1,r_2,\ldots,r_{2^n}} \quad \text{such that} D(x) = \mathsf{RE}(C, x; r_x).$$

Such a circuit D has size at least 2^n. In particular, we cannot hope to "compress" the random coins $r_1, r_2, \ldots, r_{2^n}$ into $2^{n(1-\varepsilon)}$ space, which is the target size of the obfuscated circuit. To resolve this problem, we will use a Pseudorandom Generator.

Pseudorandom generator in NC^0. A pseudorandom generator is a Boolean function $\mathsf{PRG} : \{0,1\}^n \to \{0,1\}^m$ that takes as input a uniformly random string $sd \in \{0,1\}^n$, called a *seed*, and produces a polynomially longer output $r \in \{0,1\}^m$, where $m = n^{1+\rho}$ for some constant $\rho > 0$, such that r is indistinguishable from a uniformly random m-bit string to computationally limited adversaries. Pseudorandom generators are among the most basic cryptographic primitives and have been extensively studied. Among these studies is a beautiful line of works, initiated by [61], investigating pseudorandom generators in NC^0, for which there are several candidates, including those proposed in [17, 86, 87].

Equipped with a pseudorandom generator in NC^0, we can now replace uniformly random coins $r_1, r_2, \ldots, r_{2^n}$ with pseudorandom coins expanded from a much shorter seed sd of length roughly $2^{n/(1+\rho)} = 2^{n(1-\varepsilon')}$ for some $\varepsilon' \in (0,1)$.[6] This gives a variant of the circuit D above:

$$D' = D'_{C,\mathsf{RE},\mathsf{PRG},sd} \quad \text{such that } D'(x) = \mathsf{RE}\big(C, x; \big(r_x = \mathsf{PRG}_x(sd)\big)\big),$$

where $\mathsf{PRG}(sd)$ expands to output $r_1, \ldots, r_{2^n}$ and $r_x = \mathsf{PRG}_x(sd)$ is the xth chunk of the output. Thanks to the fact that both PRG and RE are in NC^0, so is D'.

Moving forward, it suffices to devise a way to encrypt $\mathsf{spCT}(C, sd)$, from which we can expand out the truth table of D', while hiding all other information of C and sd:

$$C, sd \xrightarrow{G_{\mathsf{RE},\mathsf{PRG}}} \mathsf{TT}_{D'}, \quad \mathsf{spCT}(C, sd) \xrightarrow{\text{Expand}} \mathsf{TT}_{D'}.$$

Note that the special encryption only needs to hide (C, sd) instead of the entire description of circuit D' since RE and PRG are public algorithms.

2.3. Special encryption for NC^0 mappings

In our works [68, 69], we constructed the needed special encryption for general NC^0 mappings $G : \{0,1\}^l \to \{0,1\}^m$, where ciphertext $\mathsf{spCT}(X)$ reveals the output of G on X while hiding all other information of X, such that $\mathsf{size}(\mathsf{spCT}(X)) \sim \mathsf{size}(X) + m^{1-\delta}$ for some $\delta > 0$. In this overview, we describe half of the ideas behind our construction, which connects the special encryption with bilinear pairing groups and a new object called structured-seed pseudorandom generator. The other half of ideas is explained in Section 3.2, captured by the construction of structured-seed pseudorandom generator.

6 More precisely, the length of sd is $(2^n s^{O(1)})^{1/(1+\rho)}$ since each r_x is an $s^{O(1)}$-bit string instead of a single bit. However, the dominant term is $2^{n/(1+\rho)}$ as s is only polynomial in n.

Connection with bilinear pairing groups. As a starting point, suppose that the function G is really simple—simple enough so that it can be computed by a degree-2 polynomial mapping $Q : \mathbb{Z}_p^l \to \mathbb{Z}_p^m$ over the finite field $\mathbb{Z}_p$ for some prime modulus p. Namely,

$$\forall X \in \{0,1\}^l, \quad G(X) = Q(X),$$
$$\text{where } Q_i(X) = \left(\sum_{j,k} \alpha_{j,k} X_j \cdot X_k + \sum_k \beta_k X_k + \gamma\right) \bmod p.$$

Then, the special encryption can be implemented using bilinear pairing groups as shown in [9,75].

Bilinear pairing groups. At a high-level, pairing groups allow for computing quadratic polynomials over secret encoded values and reveal only whether the output is zero or not. More specifically, they consist of cyclic groups $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ with generators g_1, g_2, and g_T, respectively, and all of order p; $\mathbb{G}_1$ and $\mathbb{G}_2$ are referred to as the source groups and $\mathbb{G}_T$ as the target group. (In some instantiations, the two source groups are the same group, which is called symmetric pairing groups.) They support the following operations:

- (*Encode*) For every group $\mathbb{G}_i$, one can compute g_i^x for $x \in \mathbb{Z}_p$. The group element g_i^x is viewed as an *encoding* of x in the group $\mathbb{G}_i$.
- (*Group Operation*) In each group $\mathbb{G}_i$, one can perform the group operation to get $g_i^{x_1} \circ g_i^{x_2} = g_i^{x_1+x_2}$, corresponding to "homomorphic" addition modulo p in the exponent. Following the tradition of cryptography, we write the group operation multiplicatively.
- (*Bilinear Pairing*) Given two source group elements g^{x_1} and $g_2^{x_2}$, one can efficiently compute a target group element $g_T^{x_1 \cdot x_2}$, using the so-called pairing operation $e(g_1^{x_1}, g_2^{x_2}) = g_T^{x_1 \cdot x_2}$. This corresponds to "homomorphic" multiplication modulo p in the exponent. However, after multiplication, we obtain an element in the target group which cannot be paired anymore.
- (*Zero Testing*) Given a group element g_i^x, one can always tell if the encoded value is $x = 0$, by comparing the group element with the identity in G_i. Similarly, one can do equality testing to see if $g_i^x = g_i^c$ for any c.

Combining above abilities gives a "rudimentary" special encryption scheme that supports evaluation of degree-2 polynomials. A ciphertext of $X \in \mathbb{Z}_p^l$ includes encodings of every element X_l in both source groups, $((g_1^{X_1}, \dots g_1^{X_l}), (g_2^{X_1}, \dots g_2^{X_l}))$. Given these, one can "homomorphically" compute a quadratic mapping Q to obtain an encoding of the output $y = Q(X)$ in the target group $(g_T^{y_1}, \dots, g_T^{y_m})$ (without knowing the encoded input X at all); finally, if the output y happens to be a *bit string*, one can learn y in the clear via zero-testing. In summary,

$$\left((g_1^{X_1}, \dots g_1^{X_l}), (g_2^{X_1}, \dots g_2^{X_l})\right) \xrightarrow{\text{Expand}} Q(x), \quad \text{if } Q(x) \in \{0,1\}^m.$$

This fulfills the correctness of the special encryption. What about security? The ciphertext must hide all information about X beyond what is revealed by $Q(x)$, for which we resort to the security of pairing groups. For simplicity of this overview, let us gain some security intuition by assuming the strongest hardness assumptions pertaining to the pairing groups, known as the generic group model. Think of encoding as black boxes and the only way of extracting information (of the encoded values) is by performing (a combination of) the above four operations. In this model, given g^x for a secret and random $x \in \mathbb{Z}_p$, no efficient adversary can learn x. Extending further, if X were random (in $\mathbb{Z}_p$) subject to $Q(X) = y$, then the encoding would reveal only y and hide all other information of X. The only issue is that X in our example is not random—it is binary. Nevertheless, the works of [9,75] designed clever ways of *randomizing* an arbitrary input X, to a random input $\bar{X}$ subject to the only condition that an appropriate quadratic mapping $\bar{Q}$ reveals the right output $\bar{Q}(\bar{X}) = Q(X)$. Therefore, security is attained. We refer the reader to [9,75] for details about how such randomization works; in fact, it is possible to rely on much weaker assumption than the generic group model [75,96].

Challenges beyond degree 2. Unfortunately, the mapping $G_{\mathsf{RE},\mathsf{PRG}}$ we care about can only be computed by polynomials of degree much larger than 2. It is known that a Boolean function with output locality l can be computed by a multilinear degree l polynomial over $\mathbb{Z}_p$. However, the locality of Boolean PRG we need is at least 5 [86] and known randomized encodings have locality at least 4 [15].

Key idea: the preprocessing model. To overcome this challenge, our first idea is using preprocessing of inputs to help reduce the degree of computation. Instead of directly computing $G(X)$ in one shot, we separate it into two steps: First, the input is preprocessed $X' = \mathsf{pre}(X; r)$ in a randomized way using fresh random coins r, then the output is computed from the preprocessed input $y = Q(X')$. The idea is that the preprocessing can perform complex transformations on the input in order to help the computation later. The only constraint is that the preprocessing should not increase the size of its input too much, that is, $\mathsf{size}(X') \sim \mathsf{size}(X) + m^{1-\delta}$, for some $\delta > 0$. As such, it suffices to encrypt the preprocessed input $\mathsf{spCT}(X')$, from which one can recover the desired output $y = G(X)$, by evaluating a (hopefully) simpler function Q. Unfortunately, because of the restriction on the size of X', it is unclear how preprocessing alone can help.

Our second idea is to further relax the preprocessing model to allow the preprocessed input to contain a public part and a secret part $X' = (P, S)$. Importantly, the public part P should reveal no information about X to computationally limited adversaries. (In contrast, P and S together reveal X completely.) Moreover, we allow the second stage computation $Q(P, S)$ to have arbitrary constant degree in P and only restrict its degree on S to 2, that is,

$$Q_i(P, S) = \left(\sum_{j,k} \alpha_{j,k}(P) \cdot S_j \cdot S_k + \sum_k \beta_k(P) \cdot S_k + \gamma(P)\right) \bmod p,$$

where $\alpha_{j,k}, \beta_k, \gamma$ are constant-degree polynomials.

It turns out that the techniques alluded to above for special encryption for degree 2 computations can be extended (see [6, 56, 67, 96]), so that given ciphertext $\mathsf{spCT}(S)$ and P in the clear, one can homomorphically compute Q and thereby learn the output $G(X)$.

In [68, 69], we show how to compute any NC^0 Boolean function G in such a preprocessing model, assuming the Learning Parity with Noises assumption over general fields, which completes the puzzle of $i\mathcal{O}$. When applied to specific cryptographic tools, our techniques give interesting new objects. For instance, it converts any PRG in NC^0 into what we call structured-seed PRG. Given a preprocessed seed $(P, S) = \mathsf{PRG}(sd; r')$, the structured-seed PRG expands out a polynomially longer output $r = \mathsf{sPRG}(P, S)$, where the computation has only degree-2 in the private seed S, and the output r is pseudorandom given the public seed P. In the next section, we describe how to do preprocessing in the context of constructing structured-seed PRG. The same ideas can be extended to handle general NC^0 functions.

3. STRUCTURED SEED PRG

In this section we define and construct our main object, namely a structured seed PRG (sPRG). But before we do that, we introduce a few preliminaries. For any distribution $\mathcal{X}$, we denote by $x \leftarrow \mathcal{X}$ the process of sampling a value x from the distribution $\mathcal{X}$. Similarly, for a set X, we denote by $x \leftarrow X$ the process of sampling x from the uniform distribution over X. For an integer $n \in \mathbb{N}$, we denote by $[n]$ the set $\{1, \dots, n\}$. A function $\mathsf{negl} : \mathbb{N} \to \mathbb{R}$ is said to be a negligible function if for every constant $c > 0$ there exists an integer N_c such that $\mathsf{negl}(\lambda) < \lambda^{-c}$ for all $\lambda > N_c$.

Throughout, when we refer to polynomials in the security parameter, we mean constant degree polynomials that take positive value on nonnegative inputs. We denote by $\mathsf{poly}(\lambda)$ an arbitrary polynomial in λ satisfying the above requirements of nonnegativity. We denote vectors by bold letters such as $\boldsymbol{b}$ and $\boldsymbol{u}$. Matrices will be denoted by capitalized bold letters for such as $\boldsymbol{A}$ and $\boldsymbol{M}$. For any $k \in \mathbb{N}$, we denote by the tensor product $\boldsymbol{v}^{\otimes k} = \underbrace{\boldsymbol{v} \otimes \cdots \otimes \boldsymbol{v}}_{k}$ to be the standard tensor product, but converted back into a vector. This vector contains all the monomials in the variables inside $\boldsymbol{v}$ of degree exactly k.

For any two polynomials $a(\lambda, n), b(\lambda, n) : \mathbb{N} \times \mathbb{N} \to \mathbb{R}^{\geq 0}$, we say that a is polynomially smaller than b, denoted as $a \ll b$, if there exist an $\varepsilon \in (0, 1)$ and a constant $c > 0$ such that $a < b^{1-\varepsilon} \cdot \lambda^c$ for all large enough $n, \lambda \in \mathbb{N}$. The intuition behind this definition is to think of n as being a sufficiently large polynomial in λ.

Multilinear representation of polynomials and representation over $\mathbb{Z}_p$. A straightforward fact from analysis of Boolean functions is that every NC^0 function $F : \{0,1\}^n \to \{0,1\}$ can be represented by a unique constant degree multilinear polynomial $f \in \mathbb{Z}[x_1, \dots, x_n]$ that agrees with F over $\{0,1\}^n$. At times, we will also interpret $f(\boldsymbol{x})$ as a polynomial over $\mathbb{Z}_p$ for some prime p. This is done by actually reducing coefficients of f modulo p, and then evaluating the same multilinear polynomial over $\mathbb{Z}_p$. Observe that for every $\boldsymbol{x} \in \{0,1\}^n$, $f(\boldsymbol{x}) = f(\boldsymbol{x}) \bmod p$, as the for every $\boldsymbol{x} \in \{0,1\}^n$, $f(\boldsymbol{x}) \in \{0,1\}$. Furthermore, given any

NC^0 function F, finding these representations over $\mathbb{Z}$, as well as $\mathbb{Z}_p$, takes polynomial time. We now describe the notion of computational indistinguishability.

Definition 3.1 (ε-indistinguishability). We say that two ensembles $\mathcal{X} = \{\mathcal{X}_\lambda\}_{\lambda \in \mathbb{N}}$ and $\mathcal{Y} = \{\mathcal{Y}_\lambda\}_{\lambda \in \mathbb{N}}$ are ε-indistinguishable where $\varepsilon : \mathbb{N} \to [0, 1]$ if for every nonnegative polynomial $\mathsf{poly}(\cdot)$ and any adversary $\mathcal{A}$ running in time bounded by $\mathsf{poly}(\lambda)$ it holds that, for every sufficiently large $\lambda \in \mathbb{N}$,

$$\Big| \Pr_{x \leftarrow \mathcal{X}_\lambda}\big[\mathcal{A}(1^\lambda, x) = 1\big] - \Pr_{y \leftarrow \mathcal{Y}_\lambda}\big[\mathcal{A}(1^\lambda, y) = 1\big]\Big| \leq \varepsilon(\lambda).$$

We say that two ensembles are indistinguishable if they are ε-indistinguishable for some ε that is a negligible function, and subexponentially indistinguishable if they are ε-indistinguishable for $\varepsilon(\lambda) = 2^{-\lambda^c}$ for some positive constant c.

We now formally define our LPN assumption [11,29,35,66].

Definition 3.2 (δ-LPN assumption, [11,29,35,66]). Let $\delta \in (0, 1)$. We say that the δ-LPN assumption is true if the following holds: For any constant $\eta_p > 0$, any function $p : \mathbb{N} \to \mathbb{N}$ such that, for every $\ell \in \mathbb{N}$, $p(\ell)$ is a prime of ℓ^{η_p} bits, any constant $\eta_n > 0$, we set $p = p(\ell)$, $n = n(\ell) = \ell^{\eta_n}$, and $r = r(\ell) = \ell^{-\delta}$, and we require that the following two distributions are computationally indistinguishable:

$$\big\{(\boldsymbol{A}, \boldsymbol{b} = \boldsymbol{s} \cdot \boldsymbol{A} + \boldsymbol{e}) \mid \boldsymbol{A} \leftarrow \mathbb{Z}_p^{\ell \times n}, \boldsymbol{s} \leftarrow \mathbb{Z}_p^{1 \times \ell}, \boldsymbol{e} \leftarrow \mathcal{D}_r^{1 \times n}(p)\big\}_{\ell \in \mathbb{N}},$$
$$\big\{(\boldsymbol{A}, \boldsymbol{u}) \mid \boldsymbol{A} \leftarrow \mathbb{Z}_p^{\ell \times n}, \boldsymbol{u} \leftarrow \mathbb{Z}_p^{1 \times n}\big\}_{\ell \in \mathbb{N}}.$$

In addition, we say that subexponential δ-LPN holds if the two distributions above are subexponentially indistinguishable.

We now define the notion of an sPRG.

3.1. Definition of structured-seed PRG

Definition 3.3 (Syntax of structured-seed pseudorandom generators (sPRG)). Let $\tau > 1$. A structured-seed Boolean PRG (sPRG) with polynomial stretch τ is defined by the following PPT algorithms:

- $\mathsf{IdSamp}(1^n, p)$ takes as input the input length parameter n and a prime p. It samples a function index I.
- $\mathsf{SdSamp}(I)$ jointly samples two strings, a public seed and a private seed, $sd = (P, S)$, which are both vectors of dimension $\ell_{sd} = O(n)$ over $\mathbb{Z}_p$.
- $\mathsf{Eval}(I, sd)$ computes a string in $\{0, 1\}^m$. Here $m = n^\tau$.

Looking ahead, the prime p, that we choose, is set to the order of the bilinear group which has a bit length of $n^{\Theta(1)}$.

Definition 3.4 (Security of sPRG). A structured-seed Boolean PRG, sPRG, satisfies the security requirement if, for any constant $\rho > 0$, any function $p : \mathbb{N} \to \mathbb{Z}$ that takes as input a number $k \in \mathbb{N}$ and outputs a k^ρ bit prime $p(k)$, any $n \in \mathbb{N}$, with probability $1 - o(1)$ over $\mathsf{IdSamp}(1^n, p = p(n)) \to I$, it holds that the following distributions are $\varepsilon(n)$ indistinguishable:

$$\{I, P, \mathsf{Eval}(I, P, S) \mid I \leftarrow \mathsf{IdSamp}(1^n, p), sd \leftarrow \mathsf{SdSamp}(I)\},$$
$$\{I, P, \boldsymbol{r} \mid I \leftarrow \mathsf{IdSamp}(1^n, p), sd \leftarrow \mathsf{SdSamp}(I), \boldsymbol{r} \leftarrow \{0,1\}^{m(n)}\},$$

where $\varepsilon(n)$ is a negligible function. Further, we say that sPRG is subexponentially secure if $\varepsilon(n) = 2^{-n^{\Omega(1)}}$.

Definition 3.5 (Complexity and degree of sPRG). Let $\rho > 0$, $d_1, d_2 \in \mathbb{N}$ be any constants, and $p : \mathbb{N} \to \mathbb{Z}$ be any function that maps an integer k into a k^ρ bit prime $p(k)$. An sPRG has degree d_1 in public seed P and degree d_2 in S over $\mathbb{Z}_p$, denoted as $\mathsf{sPRG} \in (\deg d_1, \deg d_2)$, if for every I in the support of $\mathsf{IdSamp}(1^n, p = p(n))$, there exist efficiently generatable polynomials $g_{I,1}, \ldots, g_{I,m}$ over $\mathbb{Z}_p$ such that:

- $\mathsf{Eval}(I, (P, S)) = (g_{I,1}(P, S), \ldots, g_{I,m}(P, S))$, and
- the maximum degree of each $g_{I,j}$ over P is d_1, while the maximum degree of $g_{I,j}$ over S is d_2.

We remark that the above definition generalizes the standard notion of families of PRGs in two aspects: (1) the seed consists of a public and a private parts, jointly sampled and arbitrarily correlated, and (2) the seed may not be uniform. Therefore, we obtain the standard notion as a special case.

Definition 3.6 (Pseudorandom generators, degree, and locality). A (uniform-seed) Boolean PRG (PRG) is an sPRG with a seed sampling algorithm $\mathsf{SdSamp}(I)$ that outputs a public seed P that is an empty string and a uniformly random private seed $S \leftarrow \{0,1\}^n$. Let $d, c \in \mathbb{N}$. The PRG has multilinear degree d if, for every $n \in \mathbb{N}$ and I in the support of $\mathsf{IdSamp}(1^n)$, we have that $\mathsf{Eval}(I, sd)$ can be written as an $m(n)$-tuple of degree-d polynomials over $\mathbb{Z}$ in S. It has constant locality c if, for every $n \in \mathbb{N}$ and I in the support of $\mathsf{IdSamp}(1^n)$, every output bit of $\mathsf{Eval}(I, sd)$ depends on at most c bits of S.

In what follows next we will construct an sPRG from the LPN assumption and the existence of PRG in NC^0. For the ease of exposition, we will actually use Goldreich's PRG candidate [61] which is the most well-known conjectured PRG in NC^0.

Definition 3.7 (Goldreich's PRG). A Goldreich PRG of locality c mapping n bits to m bits is described using a predicate $f : \{0,1\}^c \to \{0,1\}$ and a hypergraph $H = \{Q_1, \ldots, Q_m\}$ where each Q_i is a randomly chosen ordered subset of $[n]$ of size c. The index I consists of f and H. Further, on input $\boldsymbol{x} \in \{0,1\}^n$, $\mathsf{PRG.Eval}(I, \boldsymbol{x}) = \boldsymbol{y}$, where $\boldsymbol{y} = (y_1, \ldots, y_m)$. Here each $y_i = f(x_{i_1}, \ldots x_{i_c})$ where $Q_i = (i_1, \ldots, i_c)$.

Remark 3.1. In a Goldreich's PRG, when the hypergraph is randomly chosen, with probability $\frac{1}{n^{O(1)}}$ it fails to be an expander. With probability $1 - o(1)$, the hypergraph has appropriate expansion properties. Under such conditions, the security of Goldreich PRGs has been very well studied [10, 12, 13, 16, 17, 30, 45, 48, 49, 61, 73, 86, 87] and is widely believed to hold. This is the precise reason in the security definition, we require pseudorandomness to hold with probability $1 - o(1)$ over the choice of I.

3.2. Construction of structured-seed PRG

Now we show how to construct our sPRG. We prove:

Theorem 3.1. *Let $d \in \mathbb{N}$, $\delta \in (0,1)$, and $\tau > 1$ be constants. Then, assuming the following:*

- *the security of locality d Goldreich's* PRG *with stretch τ, and*
- *the δ-*LPN*-assumption,*

there exists an sPRG *with polynomial stretch in* $(\deg d, \deg 2)$. *Additionally, if both assumptions are subexponentially secure, then so is the* sPRG.

We first give an overview and then dive into the construction.

Technical overview. We start with a Goldreich PRG $\mathsf{PRG} = (\mathsf{IdSamp}, \mathsf{Eval})$ with stretch τ.

Such a PRG is associated with a d-local predicate $f : \{0,1\}^d \to \{0,1\}$. Recall now how the index sampling IdSamp works. The IdSamp algorithm on input n outputs a random hypergraph H.

We start by observing that on any input $\boldsymbol{\sigma} \in \{0,1\}^n$, $\boldsymbol{y} = \mathsf{PRG.Eval}(I, \boldsymbol{\sigma})$ can be computed by degree d multilinear polynomials over $\mathbb{Z}$ as f is a d local predicate. Our high-level strategy is to somehow "preprocess" $\boldsymbol{\sigma}$ into two vectors (P, S) of small dimension (preferably $O(n)$, but anything sublinear in m works) such that $\boldsymbol{y} \in \{0,1\}^m$ can be computed in $(\deg d, \deg 2)$. Thereby this will have an effect of transferring complexity of computation to the public input. To achieve this, we preprocess $\boldsymbol{\sigma}$ into appropriate public and private seeds (P, S) and leverage the LPN assumption over $\mathbb{Z}_p$ (which is the input to sPRG.IdSamp) to show that the seed is hidden.

Our first idea towards this is that we can "encrypt" the seed $\boldsymbol{\sigma}$ using LPN samples over $\mathbb{Z}_p$ as follows:

$$\begin{aligned} \text{Sample:} \quad & \boldsymbol{A} \leftarrow \mathbb{Z}_p^{\ell \times n}, \boldsymbol{s} \leftarrow \mathbb{Z}_p^{1 \times \ell}, \boldsymbol{e} \leftarrow \mathcal{D}_r^{1 \times n}(p), \\ \text{Add to the function index } I': \quad & \boldsymbol{A}, \\ \text{Add to public seed } P: \quad & \boldsymbol{b} = \boldsymbol{s}\boldsymbol{A} + \boldsymbol{e} + \boldsymbol{\sigma}, \end{aligned}$$

where ℓ is set to be the dimension for the LPN samples. It is set so that $\ell^{\lceil \frac{d}{2} \rceil} = n$; $\mathcal{D}_r(p)$ is a distribution that samples randomly from p with probability r, and 0 otherwise. Finally, $r = \ell^{-\delta}$.

It follows directly from LPN assumption that $(\boldsymbol{A}, \boldsymbol{b})$ is pseudorandom and hides $\boldsymbol{\sigma}$. Furthermore, due to the sparsity of LPN noises, the vector $\boldsymbol{\sigma} + \boldsymbol{e}$ differs from $\boldsymbol{\sigma}$ only at an $\ell^{-\delta}$ fraction of components—thus it is a sparsely erroneous version of the seed. For $i \in [m]$, let $f_i(\boldsymbol{\sigma})$ be the locality d, degree d polynomial over $\mathbb{Z}$ such that $y_i = f_i(\boldsymbol{\sigma})$. Then, for

$i \in [m]$, consider the following polynomial:

$$h_i(\boldsymbol{b}, \overline{\boldsymbol{s}}^{\otimes \lceil \frac{d}{2} \rceil}) = f_i(\boldsymbol{b} - \boldsymbol{s}\boldsymbol{A}), \quad \overline{\boldsymbol{s}} = \boldsymbol{s}||1.$$

Above we first interpret f_i over $\mathbb{Z}$ into the field $\mathbb{Z}_p$ by simply reducing coefficients mod p, and then compute as given. Observe that f_i is a degree d polynomial in $\boldsymbol{b}$ and $\boldsymbol{s}$, therefore its degree over $\boldsymbol{b}$ is d and over $\overline{\boldsymbol{s}}^{\otimes \lceil \frac{d}{2} \rceil}$ is two. Thus h_i is a $(\deg d, \deg 2)$ polynomial. Observe that $h_i(\boldsymbol{b}, \overline{\boldsymbol{s}}^{\otimes \lceil \frac{d}{2} \rceil}) = f_i(\boldsymbol{\sigma} + \boldsymbol{e})$. The main point of this is that if we set the polynomial map $G^{(1)} = (G_1^{(1)}, \ldots, G_m^{(1)})$ by letting each $G_i^{(1)} = h_i$, and set the private seed $S = \overline{\boldsymbol{s}}^{\otimes \lceil \frac{d}{2} \rceil}$, then

$$G^{(1)}(P, S) = \mathsf{PRG.Eval}_I(\boldsymbol{\sigma} + \boldsymbol{e}).$$

The reason $G^{(1)}$ is interesting is because $\boldsymbol{e}$ is sparse. With probability $1 - 2^{-n^{\Omega(1)}}$, it is nonzero at $O(n\ell^{-\delta})$ locations. As a consequence, for any given $i \in [m]$, $f_i(\boldsymbol{\sigma}) = f_i(\boldsymbol{\sigma} + \boldsymbol{e})$ with all but $O(\ell^{-\delta})$ probability as f_i is a d local function depending on d randomly chosen inputs. Since in the hypergraph H of the Goldreich PRG, each set Q_i is chosen independently, every output is independently error prone with probability $O(\ell^{-\delta})$. Because of this, due to Chernoff style concentration bounds, out of m outputs, with probability $1 - 2^{-n^{\Omega(1)}}$, all but $T = O(m\ell^{-\delta})$ outputs are error prone.

This gives us as a nice candidate for sPRG that satisfies almost all properties! The dimension of S and P is $O(n)$ which is sublinear in m, and it can be computed by a degree $(\deg d, \deg 2)$ polynomial $G^{(1)}$. We would be done if we could somehow force the output to be correct on all the m coordinates. For the rest of the overview, we refer to the indices $i \in [m]$ such that $f_i(\boldsymbol{\sigma}) \neq f_i(\boldsymbol{\sigma} + \boldsymbol{e})$ as bad indices/outputs.

To correct errors, we further modify the polynomial and include more preprocessed information in the private seeds. We describe a sequence of ideas that lead to the final correction method, starting with two wrong ideas that illustrate the difficulties we will overcome:

- The first wrong idea is correcting by adding the difference $\mathsf{Corr} = \boldsymbol{y} - \boldsymbol{y}'$ between the correct and erroneous outputs, $\boldsymbol{y} = \mathsf{Eval}_I(\boldsymbol{\sigma})$ and $\boldsymbol{y}' = \mathsf{Eval}_I(\boldsymbol{\sigma} + \boldsymbol{e})$; we refer to Corr as the *correction vector*. To obtain the correct output, evaluation can compute the polynomial map $G^{(1)}(\boldsymbol{b}, (\overline{\boldsymbol{s}}^{\otimes \lceil \frac{d}{2} \rceil})) + \mathsf{Corr}$. The problem is that Corr must be included in the seed, but it is as long as the output and would destroy expansion. Thus, we have to make use of the fact that Corr is sparse.

- To fix expansion, the second wrong idea is adding correction only for bad outputs, so that the seed only stores nonzero entries in Corr. Recall that Corr is sparse with at most T nonzero elements. More precisely, the jth output can be computed as $G_j^{(1)}(\boldsymbol{b}, (\overline{\boldsymbol{s}}^{\otimes \lceil \frac{d}{2} \rceil})) + \mathsf{Corr}_j$ if output j is bad and without adding Corr_j otherwise. This fixes expansion, but now the evaluation polynomial depends on the location of bad outputs. If these locations are included in public information, this would leak information of the location of LPN noises, and jeopardize security. If, on the other hand, the locations are included in the private seed, then it is unclear how to maintain the requirement that the polynomial map computes only a degree-two polynomial in the private seed.

These two wrong ideas illustrate the tension between the expansion and security of our sPRG. Our construction takes care of both, by *compressing* the correction vector Corr to be polynomially shorter than the output and stored in the seed, and *expanding* it back during evaluation in a way that is oblivious of the location of bad output bits. This is possible thanks to the sparsity of the correction vector and the allowed degree-two computation on the private seed. We first illustrate our idea with the help of a simple case.

Simple case 1: much fewer than $\sqrt{m}$ bad outputs. Suppose hypothetically that the number of bad outputs is bounded by z which is much smaller than $\sqrt{m}$. Thus, if we convert Corr into a $\sqrt{m} \times \sqrt{m}$ matrix,[7] it has low rank z. We can then factorize Corr into two matrixes $\mathbf{U}$ and $\mathbf{V}$ of dimensions $\sqrt{m} \times z$ and $z \times \sqrt{m}$, respectively, such that $\mathsf{Corr} = \mathbf{UV}$, and compute the correct output as follows:

$$\forall j \in [m], \quad G_j^{(2)}\big(\boldsymbol{b}, \big(\overline{\boldsymbol{s}}^{\otimes \lceil \frac{d}{2} \rceil}, \mathbf{U}, \mathbf{V}\big)\big) = G_j^{(1)}\big(\boldsymbol{b}, \big(\overline{\boldsymbol{s}}^{\otimes \lceil \frac{d}{2} \rceil}\big)\big) + (\mathbf{UV})_{k_j, l_j},$$

where (k_j, l_j) is the corresponding index of the output bit j in the $\sqrt{m} \times \sqrt{m}$ matrix. When $z \ll \sqrt{m}$, the matrices $\mathbf{U}, \mathbf{V}$ have $2z\sqrt{m}$ field elements, which is polynomially smaller than $m = n^\tau$. As such, $G^{(2)}$ is expanding. Moreover, observe that $G^{(2)}$ has only degree 2 in the private seed and is completely oblivious of where the bad outputs are.

While the idea above works for fewer than $\sqrt{m}$ bad outputs, it does not work for the case we are dealing with. We have $T = \Theta(m\ell^{-\delta})$ bad outputs. Nevertheless, we show that a similar idea works for this case.

***T* bad outputs.** The above method, however, cannot handle more than $\sqrt{m}$ bad outputs, whereas the actual number of bad outputs can be up to $T = \Omega(m/\ell^\delta)$, much larger than $\sqrt{m}$ since δ is an arbitrarily small constant. Consider another hypothetical case where the bad outputs are evenly spread in the following sense: suppose that if we divide the matrix Corr into m/ℓ^δ blocks, each of dimension $\ell^{\delta/2} \times \ell^{\delta/2}$, there are at most ℓ^ρ bad outputs in each block where $\rho > 0$ is a really small constant (say $\delta/10$). In this case, we can "compress" each block of Corr separately using the idea from case 1. More specifically, for every block $i \in [m/\ell^\delta]$, we factor it into $\mathbf{U}_i\mathbf{V}_i$, with dimensions $\ell^{\delta/2} \times \ell^\rho$ and $\ell^\rho \times \ell^{\delta/2}$, respectively, and correct bad outputs as follows:

$$\forall j \in [m], \quad G_j^{(2)}\big(\boldsymbol{b}, \big(\overline{\boldsymbol{s}}^{\otimes \lceil \frac{d}{2} \rceil}, (\mathbf{U}_i, \mathbf{V}_i)_{i \in [\frac{m}{\ell^\delta}]}\big)\big) = G_j^{(1)}\big(\boldsymbol{b}, \big(\overline{\boldsymbol{s}}^{\otimes \lceil \frac{d}{2} \rceil}\big)\big) + (\mathbf{U}_{i_j}\mathbf{V}_{i_j})_{k_j, l_j},$$

where i_j is the block that output j belongs to, and $(k_j, l_j) \in [\ell^{\delta/2}] \times [\ell^{\delta/2}]$ is its index within this block. We observe that $G^{(2)}$ is expanding, since each matrix $\mathbf{U}_i$ or $\mathbf{V}_i$ has $\ell^{\delta/2+\rho}$ field elements, and the total number of elements is $\ell^{\delta/2+\rho} \cdot \frac{m}{\ell^\delta}$, which is polynomially smaller than m as long as δ is positive and m is polynomially related to ℓ. Moreover, $G^{(2)}$ is oblivious of the location of bad outputs just as in case 1.

This completely solves our problem except that we need to ensure that the bad outputs are well spread out in the manner described above. Our main observation here is that this

---

7 Any injective mapping from a vector to a matrix that is efficient to compute and invert will do.

is ensured due to the fact that in a Goldreich's PRG candidate the input dependence hypergraph $Q_1, \dots, Q_m$ is randomly chosen. Therefore, once we fix the location of the nonzero errors locations inside $\boldsymbol{e}$ (where with high probability $O(n\ell^{-\delta})$ locations are nonzero), in every block of ℓ^δ output bits, each entry j is independently nonzero with probability $O(\ell^{-\delta})$. Thus, in expectation each block has a constant number of bad output bits. More so, due to the Chernoff bound, it can be seen that with probability $1 - 2^{-\ell^{\Omega(1)}}$, each has at most ℓ^ρ nonzero elements. Thus, our construction can be summarized as follows:

Step 1: Assign outputs. We partition the outputs into B buckets, via a mapping $\phi_{\mathsf{bkt}} : [m] \to [B]$. The number of buckets is set to $B = m/\ell^\delta$ and the number of elements in each bucket is set to be ℓ^δ so that they exactly form partition of m. The mapping ϕ_{bkt} simply divides m by B, and outputs the remainder. Since the error $\boldsymbol{e}$ is chosen to be from the LPN error distribution and the hypergraph H of the PRG is randomly chosen, by a Chernoff-style argument, we can show that in each bucket out of ℓ^δ output bits, at most t of them are bad, except with probability $1 - 2^{-t^{\Omega(1)}}$. We will set $t = \ell^\rho$ for a tiny constant $\rho > 0$.

Step 2: Compress the buckets. Next, we organize each bucket i into a matrix $\mathbf{M}_i$ of dimension $\ell^{\delta/2} \times \ell^{\delta/2}$ and then compute its factorization $\mathbf{M}_i = \mathbf{U}_i \mathbf{V}_i$, where $\mathbf{U}_i, \mathbf{V}_i$ are matrices of dimensions $\ell^{\delta/2} \times t$ and $t \times \ell^{\delta/2}$, respectively. To form matrix $\mathbf{M}_i$, we use another mapping $\phi_{\mathsf{ind}} : [m] \to [\ell^{\delta/2}] \times [\ell^{\delta/2}]$ to assign each output bit j to an index (k_j, l_j) in the matrix of the bucket i_j it is assigned to. This assignment must guarantee that no two output bits in the same bucket (assigned according to ϕ_{bkt}) have the same index. One such way to compute $\phi_{\mathsf{ind}}(j)$ is to divide $j \in [m]$ by B. The remainder is set as $\phi_{\mathsf{bkt}}(j)$, and the quotient is divided further by $\ell^{\delta/2}$. The quotient and the remainder from this division are set as the resulting indices (k_j, l_j). Once we have this, $(\mathbf{M}_i)_{k,l}$ is set to Corr_j if there is j such that $\phi_{\mathsf{bkt}}(j) = i$ and $\phi_{\mathsf{ind}}(j) = (k, l)$. Since every matrix $\mathbf{M}_i$ has at most t nonzero entries, we can factor them and compute the correct output as:

$$\begin{aligned} \forall j \in [m], \quad & G_j^{(2)}\big(\boldsymbol{b}, \underbrace{\big(\bar{\boldsymbol{s}}^{\otimes \lceil \frac{d}{2} \rceil}, (\mathbf{U}_i, \mathbf{V}_i)_{i \in [B]}\big)}_{S}\big) \\ & = G_j^{(1)}\big(\boldsymbol{b}, \big(\bar{\boldsymbol{s}}^{\otimes \lceil \frac{d}{2} \rceil}\big)\big) + (\mathbf{U}_{\phi_{\mathsf{bkt}}(j)} \cdot \mathbf{V}_{\phi_{\mathsf{bkt}}(j)})_{\phi_{\mathsf{ind}}(j)}, \end{aligned}$$

$G^{(2)}$ is expanding because the number of field elements in $\mathbf{U}_i$'s and $\mathbf{V}_i$'s are much smaller than m, namely $2t\ell^{\delta/2}B = O(m\ell^{-\delta/2+\rho}) \ll m$. We set $I' = (I, \boldsymbol{A}, \phi_{\mathsf{bkt}}, \phi_{\mathsf{ind}})$.

Step 3: Zeroize if uneven buckets. Finally, to deal with the low probability event that some bucket contains more than t bad outputs, we introduce a new variable called a flag. If this occurs, our sPRG sets P and S as all-zero vectors. In this case the evaluation always outputs 0. This gives us our candidate sPRG. For security, observe that the polynomial map $G^{(2)}$ is independent of the location of LPN noises. With probability $1 - 2^{-n^{\Omega(1)}}$, the evaluation results in output $\boldsymbol{y}$. Therefore, by the LPN over $\mathbb{Z}_p$ assumption, the seed $\boldsymbol{\sigma}$ of PRG is hidden and the security of PRG ensures that the output is pseudorandom when it is not all zero (which occurs with a subexponentially small probability). We now proceed to the formal construction and proof.

Construction. We now formally describe our scheme. Assume the premise of the theorem. Let (IdSamp, Eval) be the function index sampling algorithm and evaluation algorithm for the Goldreich PRG. Recall that its seed consists of only a private seed sampled uniformly at random.

We first introduce and recall some notation. The construction is parameterized by

- the input length n and output length $m = n^{\tau}$ of the Goldreich PRG (PRG),
- the stretch $\tau > 1$ and degree/locality d of the Goldreich PRG,
- the LPN secret dimension $\ell = n^{1/\lceil d/2 \rceil}$ and the error probability $r = \ell^{-\delta}$,
- a slack parameter $t = \ell^{\rho}$ used for bounding the number of bad outputs in each bucket,
- a parameter $B = m/\ell^{\delta}$ that indicates the number of buckets used, and
- a parameter $c = \ell^{\delta}$ that indicates the capacity of each bucket; it is set so that $c \cdot B = m$,
- a parameter η, which is the dimension of each bucket; we set $\eta = \sqrt{c}$,
- assignment function $\phi_{\mathsf{bkt}} : [m] \to [B]$ that is computed by dividing input j by B and returning its remainder,
- assignment function $\phi_{\mathsf{ind}} : [m] \to [\eta]$ that is computed by dividing input $j \in [m]$ by B and dividing further the quotient with η, and returning the quotient and the remainder of this division.

$I' \leftarrow \mathsf{IdSamp}'(1^{n'}, p)$: Generate the public index as follows:

- Sample $I \leftarrow \mathsf{PRG.IdSamp}(1^n)$ and $\boldsymbol{A} \leftarrow \mathbb{Z}_p^{\ell \times n}$.
- Output $I' = (I, \boldsymbol{\phi} = (\phi_{\mathsf{bkt}}, \phi_{\mathsf{ind}}), \boldsymbol{A})$.

(Note that the PRG seed length n below is an efficiently computable polynomial in n', and can be inferred from the next seed sampling algorithm. See Claim 3.1 for the exact relationship between n and n'.)

$sd \leftarrow \mathsf{SdSamp}'(I')$: Generate the seed as follows:

- Sample a PRG seed $\boldsymbol{\sigma} \leftarrow \{0, 1\}^n$.

- Prepare samples of LPN over $\mathbb{Z}_p$: Sample $\boldsymbol{s} \leftarrow \mathbb{Z}_p^{1\times\ell}$, $\boldsymbol{e} \leftarrow \mathcal{D}_r^{1\times n}(p)$, and set

$$\boldsymbol{b} = \boldsymbol{s}\boldsymbol{A} + \boldsymbol{\sigma} + \boldsymbol{e}.$$

- Find indices $i \in [n]$ of seed bits where $\boldsymbol{\sigma} + \boldsymbol{e}$ and $\boldsymbol{\sigma}$ differ, which are exactly these indices where $\boldsymbol{e}$ is not 0, and define

$$\mathsf{ERR} = \{i \mid \sigma_i + e_i \neq \sigma_i\} = \{i \mid e_i \neq 0\}.$$

  We say a seed index i is *erroneous* if $i \in \mathsf{ERR}$. Since LPN noise is sparse, errors are sparse.

- Find indices $j \in [m]$ of outputs that depend on one or more erroneous seed indices. Let Vars_j denote the indices of seed bits that the jth output of $\mathsf{Eval}_{\boldsymbol{I}}$ depends on. Define

$$\mathsf{BAD} = \left\{j \mid |\mathsf{Vars}_j \cap \mathsf{ERR}| \geq 1\right\}.$$

  We say an output index j is bad if $j \in \mathsf{BAD}$, and good otherwise.

- Set $\mathsf{flag} = 0$ if there is some bucket containing too many bad outputs: $\exists i \in [B]$, $|\phi_{\mathsf{bkt}}^{-1}(i) \cap \mathsf{BAD}| > t$. Otherwise, set $\mathsf{flag} = 1$.

- Compute the outputs of PRG on inputting the correct seed and the erroneous seed, $\boldsymbol{y} = \mathsf{PRG.Eval}_{\boldsymbol{I}}(\boldsymbol{\sigma})$ and $\boldsymbol{y}' = \mathsf{PRG.Eval}_{\boldsymbol{I}}(\boldsymbol{\sigma} + \boldsymbol{e})$. Set the correction vector $\mathsf{Corr} = \boldsymbol{y} - \boldsymbol{y}'$.

- Construct matrices $\mathbf{M}_1, \ldots, \mathbf{M}_B$ by setting

$$\forall j \in [m], \quad (\mathbf{M}_{\phi_{\mathsf{bkt}}(j)})_{\phi_{\mathsf{ind}}(j)} = \mathsf{Corr}_j.$$

  Every other entry is set to 0.

- "Compress" matrices $\mathbf{M}_1, \ldots, \mathbf{M}_B$ as follows:
  - If $\mathsf{flag} = 1$, for every $i \in [B]$, compute factorization

$$\mathbf{M}_i = \mathbf{U}_i\mathbf{V}_i, \quad \mathbf{U}_i \in \mathbb{Z}_p^{\eta\times t}, \mathbf{V}_i \in \mathbb{Z}_p^{t\times\eta}.$$

    This factorization exists because, when $\mathsf{flag} = 1$, each bucket has at most t nonzero entries, and therefore its rank is less than or equal to t.

  - If $\mathsf{flag} = 0$, for every $i \in [B]$, set $\mathbf{U}_i$ and $\mathbf{V}_i$ to be 0 matrices.

- Set the public seed to

$$P = (\boldsymbol{b} \cdot \mathsf{flag}).$$

  This means that, if $\mathsf{flag} = 0$, P is the all-zero vector in $\mathbb{Z}_p^n$.

- Prepare the private seed S as follows. Let $\overline{s} = s||1$ and set

$$S = \left(\mathsf{flag} \cdot \overline{s}^{\otimes \lceil \frac{d}{2} \rceil}, \{\mathbf{U}_i, \mathbf{V}_i\}_{i \in [B]}\right). \tag{3.1}$$

This means that, if $\mathsf{flag} = 0$, S is the all-zero vector over $\mathbb{Z}_p$. Output $sd = (P, S)$ as $\mathbb{Z}_p$ elements.

$y \to \mathsf{Eval}'(I', sd)$: Compute $y \leftarrow \mathsf{Eval}(I, \sigma)$ and output $z = \mathsf{flag} \cdot y$. Looking ahead, $\mathsf{flag} = 1$ will happen with all but subexponentially small probability. This computation is done via a polynomial map $G^{(2)}$:

- Every output bit of Eval is a linear combination of degree d monomials (without loss of generality, assume that all monomials have exactly degree d which can be done by including 1 in the seed σ).

  Notation. Let us introduce some notation for monomials. A monomial h on a vector a is represented by the set of indices $h = \{i_1, i_2, \ldots, i_k\}$ of variables used in it; h evaluated on a is $\prod_{i \in h} a_i$ if $h \neq \emptyset$ and 1 otherwise. We will use the notation $a_h = \prod_{i \in h} a_i$. We abuse notation to also use a polynomial g to denote the set of monomials involved in its computation; hence $h \in g$ says monomial h has a nonzero coefficient in g.

  With the above notation, we can write Eval as

$$\forall j \in [m], \quad y_j = \mathsf{Eval}_j(\sigma) = L_j((\sigma_h)_{h \in \mathsf{Eval}_j}), \quad \text{for a linear } L_j.$$

- $(A, b = sA + x)$ in the public seed encodes $x = \sigma + e$. Therefore, we can compute every monomial x_v as follows:

$$x_i = \langle c_i, \overline{s} \rangle, \quad c_i = -a_i^{\mathrm{T}}||b_i, a_i \text{ is the } i\text{th column of } A,$$
$$x_v = \langle \otimes_{i \in v} c_i, \otimes_{i \in v} \overline{s} \rangle.$$

  (Recall that $\otimes_{i \in v} z_i = z_{i_1} \otimes \cdots \otimes z_{i_k}$ if $v = \{i_1, \ldots, i_k\}$ and is not empty; otherwise, it equals 1.) Combining with the previous step, we obtain a polynomial $G^{(1)}(b, S)$ that computes $\mathsf{Eval}(\sigma + e)$:

$$G_j^{(1)}(b, S) := L_j\left(\left(\langle \otimes_{i \in v} c_i, \otimes_{i \in v} \overline{s} \rangle\right)_{v \in \mathsf{Eval}_j}\right). \tag{3.2}$$

  Note that $G^{(1)}$ implicitly depends on A contained in I'. Since all relevant monomials v have degree d, we have that $G^{(1)}$ has degree at most d in P, and degree 2 in S. The latter follows from the fact that S contains $\overline{s}^{\otimes \lceil \frac{d}{2} \rceil}$, and hence $S \otimes S$ contains all monomials in s of total degrees d.

  Since only bad outputs depend on erroneous seed bits such that $\sigma_i + e_i \neq \sigma_i$, we have that the output of $G^{(1)}$ agrees with the correct output $y = \mathsf{Eval}(\sigma)$ on all good output bits:

$$\forall j \notin \mathsf{BAD}, \quad \mathsf{Eval}_j(\sigma) = G_j^{(1)}(b, S).$$

- To further correct bad output bits, we add to $G^{(1)}$ all the expanded correction vectors as follows:

$$G_j^{(2)}(P,S) := G_j^{(1)}(\boldsymbol{b},S) + \big(\mathbf{U}_{\phi_{\mathsf{bkt}}(j)}\mathbf{V}_{\phi_{\mathsf{bkt}}}(j)\big)_{\phi_{\mathsf{ind}}(j)}$$
$$= G_j^{(1)}(\boldsymbol{b},S) + (\mathbf{M}_{\phi_{\mathsf{bkt}}(j)})_{\phi_{\mathsf{ind}}(j)}.$$

  We have that $G^{(2)}$ agrees with the correct output $\boldsymbol{y} = \mathsf{Eval}(\boldsymbol{\sigma})$ if $\mathsf{flag} = 1$. This is because, under the condition for $\mathsf{flag} = 1$, every entry j in the correction vector Corr_j is placed at entry $(\mathbf{M}_{\phi_{\mathsf{bkt}}(j)})_{\phi_{\mathsf{ind}}(j)}$. Adding it back as above produces the correct output. Observe that the function is quadratic in S and degree d in the public component of the seed P.

- When $\mathsf{flag} = 0$, however, sPRG needs to output all zero. This happens because P and S are both all-zero vectors and $G^{(2)}$ does not use a constant term. At last, $G^{(2)}$ has degree d in the public seed, and only degree 2 in the private seed, as desired.

Analysis of stretch. We derive a set of constraints, under which sPRG has polynomial stretch. Recall that PRG output length is $m = n^{\tau}$, degree d, LPN secret dimension $\ell = n^{1/\lceil d/2 \rceil}$, modulus p is a prime, and the slack parameter $t = \ell^{\rho}$.

Claim 3.1. *For the parameters as set in the Construction,* sPRG *has stretch of* τ' *for some constant* $\tau' > 1$.

Proof. Let us start by analyzing the dimension of the public and private seeds:

- The public seed contains $P = (\boldsymbol{b} \cdot \mathsf{flag})$ and has $O(n)$ field elements.
- The private seed S contains S_1, S_2 as follows:

$$S_1 = \mathsf{flag} \cdot \big(\overline{\boldsymbol{s}}^{\otimes \lceil \frac{d}{2} \rceil}\big), \quad S_2 = \{\mathbf{U}_i, \mathbf{V}_i\}_{i \in [B]}.$$

  The dimension of S_1 is $O(n)$ as $\ell = n^{\frac{1}{\lceil \frac{d}{2} \rceil}}$, and S_2 consists of $O(B \cdot \eta \cdot t)$ field elements. This consist of $O(m\ell^{-(\delta/2-\rho)})$ field elements. Because $m = n^{\tau}$ and $\ell = n^{\frac{1}{\lceil \frac{d}{2} \rceil}}$, we have:

$$\mathsf{dim}(S_1) = O(n),$$
$$\mathsf{dim}(S_2) = O\Big(n^{\tau - \frac{(\delta/2-\rho)}{\lceil \frac{d}{2} \rceil}}\Big),$$
$$\mathsf{dim}(P,S) = O\Big(n + n^{\tau - \frac{2\delta}{5 \cdot \lceil \frac{d}{2} \rceil}}\Big).$$

The last equality uses $\rho = \delta/10$. We set $n' = n + n^{\tau - \frac{2\delta}{5\lceil \frac{d}{2} \rceil}}$, and therefore $m = n'^{\tau'}$ for some $\tau' > 1$. This concludes the proof. ■

Proof of pseudorandomness. We prove the following proposition which implies that sPRG is secure.

Proposition 3.1. *Let $\delta > 0$, $\tau > 1$, $d \in \mathbb{N}$, and $\beta \geq 0$ be constants. Assume the following assumptions hold:*

- δ-LPN, *and*
- PRG *is a secure Goldreich PRG with stretch τ and locality d.*

Then, for any prime generating function $p : \mathbb{N} \to \mathbb{N}$ that takes as input an integer k and outputs a prime p of bit length k^β, we have the following. Let $n \in \mathbb{N}$ and $p = p(n)$, then it holds that, with probability $1 - o(1)$ over $I \leftarrow \mathsf{IdSamp}(1^n)$,

$$\left\{(I', P, \boldsymbol{z}) : \boldsymbol{A} \leftarrow \mathbb{Z}_p^{\ell \times n}, I' = (I, \boldsymbol{\phi}, \boldsymbol{A}, p), (P, S) \leftarrow \mathsf{SdSamp}'(I'), \boldsymbol{z} \leftarrow \mathsf{Eval}'(I, sd)\right\},$$
$$\left\{(I', P, \boldsymbol{r}) : \boldsymbol{A} \leftarrow \mathbb{Z}_p^{\ell \times n}, I' = (I, \boldsymbol{\phi}, \boldsymbol{A}, p), (P, S) \leftarrow \mathsf{SdSamp}'(I'), \boldsymbol{r} \leftarrow \{0, 1\}^m\right\},$$

are computationally indistinguishable. Further assuming that the assumptions are subexponentially secure, these distributions are subexponentially indistinguishable.

We first recall the structure of P and the evaluation $\boldsymbol{z}$: P consists of $\mathsf{flag} \cdot P$, where $\boldsymbol{b} = \boldsymbol{s}\boldsymbol{A} + \boldsymbol{e} + \boldsymbol{\sigma}$ and $\boldsymbol{\sigma}$ is a randomly generated PRG seed. As shown in the correctness proof, $\boldsymbol{z} = \mathsf{flag} \cdot \boldsymbol{y}$, where $\boldsymbol{y} = \mathsf{Eval}(I, \boldsymbol{\sigma})$. If flag is always equal to 1, the proof becomes trivial: P is pseudorandom due to LPN assumption and therefore it computationally hides $\boldsymbol{\sigma}$. Secondly, once $\boldsymbol{\sigma}$ is hidden with probability $1 - o(1)$ over choice of I, $\boldsymbol{y} = \mathsf{Eval}(I, \boldsymbol{\sigma})$ is computationally indistinguishable to a random string $\boldsymbol{r}$. We observe that $\mathsf{flag} = 1$, with all but $2^{-n^{\Omega(1)}}$ probability, by the random choice of hypergraph underlying the PRG. We thus have (proof omitted, see [68] for details):

Lemma 3.2. *In the* sPRG *construction,* $\Pr[\mathsf{flag} = 1] = 1 - 2^{-n^{\Omega(1)}}$.

We now list a few hybrid experiments, $\mathsf{H}_0, \mathsf{H}_1, \mathsf{H}_2$, and H_3, where the first corresponds to the first distribution in the proposition, and the last corresponds to the second distribution in the proposition. We abuse notation to also use H_i to denote the output distribution of the hybrid.

Hybrid H_0 samples $(I', P, \boldsymbol{y})$ honestly as in the first distribution, that is,

$$\begin{aligned}
\text{Sample:} \quad & \boldsymbol{A} \leftarrow \mathbb{Z}_p^{\ell \times n}, \boldsymbol{s} \leftarrow \mathbb{Z}_p^{1 \times \ell}, \boldsymbol{e} \leftarrow \mathcal{D}_r^{1 \times n}(p), \boldsymbol{\sigma} \leftarrow \{0,1\}^n \\
& \boldsymbol{b} = \boldsymbol{s}\boldsymbol{A} + \boldsymbol{e} + \boldsymbol{\sigma}, I \leftarrow \mathsf{IdSamp}(1^n), \boldsymbol{y} = \mathsf{Eval}_I(\boldsymbol{\sigma}) \\
\text{Output:} \quad & I, \boldsymbol{\phi}, \boldsymbol{A}, P = \mathsf{flag} \cdot \boldsymbol{b}, \mathsf{flag} \cdot \boldsymbol{y} \\
& \text{where } \mathsf{flag} = 1 \text{ iff:} \\
& \forall i \in [B], \left|\phi_{\mathsf{bkt}}^{-1}(i) \cap \mathsf{BAD}\right| \leq \ell^\rho
\end{aligned}$$

Hybrid H_1 computes the distribution as before, except that we set $\mathsf{flag} = 1$:

$$\begin{aligned} \text{Sample:} \quad & \boldsymbol{A} \leftarrow \mathbb{Z}_p^{\ell \times n}, \boldsymbol{s} \leftarrow \mathbb{Z}_p^{1 \times \ell}, \boldsymbol{e} \leftarrow \mathcal{D}_r^{1 \times n}(p), \boldsymbol{\sigma} \leftarrow \{0,1\}^n \\ & \boldsymbol{b} = \boldsymbol{s}\boldsymbol{A} + \boldsymbol{e} + \boldsymbol{\sigma}, I \leftarrow \mathsf{IdSamp}(1^n), \boldsymbol{y} = \mathsf{Eval}_I(\boldsymbol{\sigma}) \\ \text{Output:} \quad & I, \boldsymbol{\phi}, \boldsymbol{A}, P = \boldsymbol{b}, \boldsymbol{y} \end{aligned}$$

Note that Hybrid H_0 and Hybrid H_1 are statistically indistinguishable with statistical distance $2^{-n^{\Omega(1)}}$ due to Lemma 3.2.

Hybrid H_2 computes $\boldsymbol{b}$ by sampling it as a random vector over $\mathbb{Z}_p$:

$$\begin{aligned} \text{Sample:} \quad & \boldsymbol{A} \leftarrow \mathbb{Z}_p^{\ell \times n}, \boldsymbol{\sigma} \leftarrow \{0,1\}^n \\ & \boldsymbol{b} \leftarrow \mathbb{Z}_p^n, I \leftarrow \mathsf{IdSamp}(1^n), \boldsymbol{y} = \mathsf{Eval}_I(\boldsymbol{\sigma}) \\ \text{Output:} \quad & I, \boldsymbol{\phi}, \boldsymbol{A}, P = \boldsymbol{b}, \boldsymbol{y} \end{aligned}$$

Note that Hybrid H_1 and Hybrid H_2 are computationally indistinguishable due to the security of the LPN assumption. The only difference between the hybrids is how $\boldsymbol{b}$ is generated. In Hybrid H_1, $\boldsymbol{b}$ is generated by sampling $\boldsymbol{s}$ and computing $\boldsymbol{b} = \boldsymbol{s}\boldsymbol{A} + \boldsymbol{e} + \boldsymbol{\sigma}$ where $\boldsymbol{e}$ is generated using LPN error distribution, and in Hybrid H_2 it is generated by sampling $\boldsymbol{b}$ by first sampling a uniform vector $\boldsymbol{u}$ and then adding $\boldsymbol{\sigma}$ (which is equivalent to just sampling $\boldsymbol{b}$ uniformly). Note that $\boldsymbol{e}$ and $\boldsymbol{s}$ appear nowhere else in the hybrids. Thus, relying on a straightforward reduction, one can reduce indistinguishability of these hybrids to the security of LPN. Further, if LPN is subexponentially secure, then these hybrids are subexponentially indistinguishable.

Hybrid H_3 simply replaces $\boldsymbol{y}$ by sampling it as a random vector in $\{0,1\}^m$:

$$\begin{aligned} \text{Sample:} \quad & \boldsymbol{A} \leftarrow \mathbb{Z}_p^{\ell \times n} \\ & \boldsymbol{b} \leftarrow \mathbb{Z}_p^n, I \leftarrow \mathsf{IdSamp}(1^n), \boldsymbol{y} \leftarrow \{0,1\}^m \\ \text{Output:} \quad & I, \boldsymbol{\phi}, \boldsymbol{A}, P = \boldsymbol{b}, \boldsymbol{y} \end{aligned}$$

We now show the following claim:

Claim 3.2. *Assuming* PRG *is a secure Goldreich's PRG, then, with probability* $1 - o(1)$ *over* I*, for any probabilistic polynomial time adversary* $\mathcal{A}$*,*

$$\left|\Pr\left[\mathcal{A}(\mathsf{H}_2) = 1\right] - \Pr\left[\mathcal{A}(\mathsf{H}_3) = 1\right]\right| \leq \varepsilon_{\mathsf{PRG}}(n),$$

where $\varepsilon_{\mathsf{PRG}}$ *is the distinguishing advantage of the* PRG.

We prove it by contradiction. Assume that for $1 - \Omega(1)$ probability over the choice of I,

$$\left|\Pr\left[\mathcal{A}(\mathsf{H}_2) = 1\right] - \Pr\left[\mathcal{A}(\mathsf{H}_3) = 1\right]\right| > \varepsilon_{\mathsf{PRG}}(n).$$

We will show that if this happens then there exists a polynomial time distinguisher D, for which with probability $1-\Omega(1)$ over $I \leftarrow \mathsf{IdSamp}(1^n)$,

$$\big|\Pr\big[D\big(I, \mathsf{Eval}\big(I, \boldsymbol{\sigma} \leftarrow \{0,1\}^n\big)\big) = 1\big] \\ - \Pr\big[D(I, \mathsf{Eval}\big(I, \boldsymbol{r} \leftarrow \{0,1\}^m\big) = 1\big]\big| > \varepsilon_{\mathsf{PRG}}(n),$$

thereby breaking the PRG security. We show this by building D as a reduction relying on $\mathcal{A}$. The reduction gets as input an index I, $\boldsymbol{y}$ from the PRG challenger, and samples $\boldsymbol{A} \leftarrow \mathbb{Z}_p^{n\times\ell}$ and $\boldsymbol{b} \leftarrow \mathbb{Z}_p^n$. It sends to $\mathcal{A}$ the input $(I, \boldsymbol{\phi}, \boldsymbol{A}, P = \boldsymbol{b}, \boldsymbol{y})$ and outputs whatever it outputs. Note that the view of D is identical to the adversary for the PRG game. For $\mathcal{A}$, if $\boldsymbol{y}$ is generated using PRG.Eval, then its view is identical to Hybrid H_2, otherwise it is identical to Hybrid H_3. Therefore, if $\mathcal{A}$ manages to distinguish between the hybrids with probability more than $\varepsilon_{\mathsf{PRG}}$ over $1-o(1)$ choice of I, then D will also be able to win the PRG game security with probability more than $\varepsilon_{\mathsf{PRG}}$. Thus, the claim follows.

FUNDING

Aayush Jain was partially supported by grants listed under Amit Sahai, and a Google PhD fellowship. Huijia Lin was supported by NSF grants CNS-1528178, CNS-1929901, CNS-1936825 (CAREER), CNS-2026774, a Hellman Fellowship, a JP Morgan AI research Award, DARPA and ARO Contract No. W911NF-15-C-0236, and a subcontract No. 2017-002 through Galois. Amit Sahai was supported in part from DARPA SAFEWARE and SIEVE awards, NTT Research, NSF Frontier Award 1413955, and NSF grant 1619348, BSF grant 2012378, a Xerox Faculty Research Award, a Google Faculty Research Award, and an Okawa Foundation Research Grant. This material is based upon work supported by the Defense Advanced Research Projects Agency through Award HR00112020024 and the ARL under Contract W911NF-15-C- 0205. The views expressed are those of the authors and do not reflect the official policy or position of the Department of Defense, DARPA, ARO, Simons, Intel, Okawa Foundation, ODNI, IARPA, DIMACS, BSF, Xerox, the National Science Foundation, NTT Research, Google, or the US Government.

REFERENCES

[1] S. Agrawal, Indistinguishability obfuscation without multilinear maps: New methods for bootstrapping and instantiation. In *Eurocrypt 2019, part I*, edited by Y. Ishai and V. Rijmen, pp. 191–225, Lecture Notes in Comput. Sci. 11476, Springer, Heidelberg, 2019.

[2] S. Agrawal and A. Pellet-Mary, Indistinguishability obfuscation without maps: Attacks and fixes for noisy linear FE. In *Eurocrypt 2020, part I*, edited by V. Rijmen and Y. Ishai, pp. 110–140, Lecture Notes in Comput. Sci., Springer, Heidelberg, 2020.

[3] M. R. Albrecht, P. Farshim, D. Hofheinz, E. Larraia, and K. G. Paterson, Multilinear maps from obfuscation. In *TCC 2016-a, part I*, edited by E. Kushilevitz and T. Malkin, pp. 446–473, Lecture Notes in Comput. Sci. 9562, Springer, Heidelberg, 2016.

[4] M. Alekhnovich, More on average case vs approximation complexity. In *44th FOCS*, pp. 298–307, IEEE Computer Society Press, 2003.

[5] P. Ananth and A. Jain, Indistinguishability obfuscation from compact functional encryption. In *Crypto 2015, part I*, edited by R. Gennaro and M. J. B. Robshaw, pp. 308–326, Lecture Notes in Comput. Sci. 9215, Springer, Heidelberg, 2015.

[6] P. Ananth, A. Jain, H. Lin, C. Matt, and A. Sahai, Indistinguishability obfuscation without multilinear maps: New paradigms via low degree weak pseudorandomness and security amplification. In *Crypto 2019, part III*, edited by A. Boldyreva and D. Micciancio, pp. 284–332, Lecture Notes in Comput. Sci. 11694, Springer, Heidelberg, 2019.

[7] P. Ananth, A. Jain, and A. Sahai, Indistinguishability obfuscation from functional encryption for simple functions. *IACR Cryptol. ePrint Arch.* **730** (2015), 2015.

[8] P. Ananth, A. Jain, and A. Sahai, Indistinguishability obfuscation without multilinear maps: iO from LWE, bilinear maps, and weak pseudorandomness. *IACR Cryptol. ePrint Arch.* **2018** (2018), 615.

[9] P. Ananth and A. Sahai, Projective arithmetic functional encryption and indistinguishability obfuscation from degree-5 multilinear maps. In *Eurocrypt 2017, part I*, edited by J. Coron and J. B. Nielsen, pp. 152–181, Lecture Notes in Comput. Sci. 10210, Springer, Heidelberg, 2017.

[10] B. Applebaum, Pseudorandom generators with long stretch and low locality from random local one-way functions. *SIAM J. Comput.* **42** (2013), no. 5, 2008–2037.

[11] B. Applebaum, J. Avron, and C. Brzuska, Arithmetic cryptography: extended abstract. In *ITCS 2015*, edited by T. Roughgarden, pp. 143–151, ACM Press, 2015.

[12] B. Applebaum, B. Barak, and A. Wigderson, Public-key cryptography from different assumptions. In *42nd ACM STOC*, edited by L. J. Schulman, pp. 171–180, ACM Press, 2010.

[13] B. Applebaum, A. Bogdanov, and A. Rosen, A dichotomy for local small-bias generators. In *TCC 2012*, edited by R. Cramer, pp. 600–617, Lecture Notes in Comput. Sci. 7194, Springer, Heidelberg, 2012.

[14] B. Applebaum, I. Damgård, Y. Ishai, M. Nielsen, and L. Zichron, Secure arithmetic computation with constant computational overhead. In *Crypto 2017, part I*, edited by J. Katz and H. Shacham, pp. 223–254, Lecture Notes in Comput. Sci. 10401, Springer, Heidelberg, 2017.

[15] B. Applebaum, Y. Ishai, and E. Kushilevitz, Cryptography in NC^0. In *FOCS*, pp. 166–175, IEEE Computer Society, 2004.

[16] B. Applebaum and E. Kachlon, Sampling graphs without forbidden subgraphs and unbalanced expanders with negligible error. In *60th FOCS*, edited by D. Zuckerman, pp. 171–179, IEEE Computer Society Press, 2019.

[17] B. Applebaum and S. Lovett, Algebraic attacks against random local functions and their countermeasures. In *48th ACM STOC*, edited by D. Wichs and Y. Mansour, pp. 1087–1100, ACM Press, 2016.

[18] S. Badrinarayanan, E. Miles, A. Sahai, and M. Zhandry, Post-zeroizing obfuscation: new mathematical tools, and the case of evasive circuits. In *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 II*, pp. 764–791, Lecture Notes in Comput. Sci. 9666, Springer, 2016.

[19] B. Barak, Z. Brakerski, I. Komargodski, and P. K. Kothari, Limits on low-degree pseudorandom generators (or: Sum-of-squares meets program obfuscation). In *Eurocrypt 2018, part II*, edited by J. B. Nielsen and V. Rijmen, pp. 649–679, Lecture Notes in Comput. Sci. 10821, Springer, Heidelberg, 2018.

[20] B. Barak, S. Garg, Y. T. Kalai, O. Paneth, and A. Sahai, Protecting obfuscation against algebraic attacks. In *Eurocrypt 2014*, edited by P. Q. Nguyen and E. Oswald, pp. 221–238, Lecture Notes in Comput. Sci. 8441, Springer, Heidelberg, 2014.

[21] B. Barak, S. B. Hopkins, A. Jain, P. Kothari, and A. Sahai, Sum-of-squares meets program obfuscation, revisited. In *Eurocrypt 2019, part I*, edited by Y. Ishai and V. Rijmen, pp. 226–250, Lecture Notes in Comput. Sci. 11476, Springer, Heidelberg, 2019.

[22] B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S. P. Vadhan, and K. Yang, On the (im)possibility of obfuscating programs. In *Crypto 2001*, edited by J. Kilian, pp. 1–18, Lecture Notes in Comput. Sci. 2139, Springer, Heidelberg, 2001.

[23] J. Bartusek, Y. Ishai, A. Jain, F. Ma, A. Sahai, and M. Zhandry, Affine determinant programs: a framework for obfuscation and witness encryption. In *ITCS 2020*, edited by T. Vidick, LIPIcs. Leibniz Int. Proc. Inform. 151, pp. 82:1–82:39, 2020.

[24] J. Bartusek and G. Malavolta, Candidate obfuscation of null quantum circuits and witness encryption for QMA. *IACR Cryptol. ePrint Arch.* **2021** (2021), 421.

[25] A. Bishop, A. Jain, and L. Kowalczyk, Function-hiding inner product encryption. In *Asiacrypt 2015, part I*, edited by T. Iwata and J. H. Cheon, pp. 470–491, Lecture Notes in Comput. Sci. 9452, Springer, Heidelberg, 2015.

[26] N. Bitansky, R. Nishimaki, A. Passelègue, and D. Wichs, From cryptomania to obfustopia through secret-key functional encryption. In *TCC 2016-b, part II*, edited by M. Hirt and A. D. Smith, pp. 391–418, Lecture Notes in Comput. Sci. 9986, Springer, Heidelberg, 2016.

[27] N. Bitansky, O. Paneth, and A. Rosen, On the cryptographic hardness of finding a Nash equilibrium. In *56th FOCS*, edited by V. Guruswami, pp. 1480–1498, IEEE Computer Society Press, 2015.

[28] N. Bitansky and V. Vaikuntanathan, Indistinguishability obfuscation from functional encryption. In *56th FOCS*, edited by V. Guruswami, pp. 171–190, IEEE Computer Society Press, 2015.

[29] A. Blum, M. L. Furst, M. J. Kearns, and R. J. Lipton, Cryptographic primitives based on hard learning problems. In *Crypto'93*, edited by D. R. Stinson, pp. 278–291, Lecture Notes in Comput. Sci. 773, Springer, Heidelberg, 1994.

[30] A. Bogdanov and Y. Qiao, On the security of Goldreich's one-way function. *Comput. Complexity* **21** (2012), no. 1, 83–127.

[31] D. Boneh, X. Boyen, and H. Shacham, Short group signatures. In *Crypto 2004*, edited by M. Franklin, pp. 41–55, Lecture Notes in Comput. Sci. 3152, Springer, Heidelberg, 2004.

[32] D. Boneh, A. Sahai, and B. Waters, Functional encryption: definitions and challenges. In *TCC 2011*, edited by Y. Ishai, pp. 253–273, Lecture Notes in Comput. Sci. 6597, Springer, Heidelberg, 2011.

[33] D. Boneh and M. Zhandry, Multiparty key exchange, efficient traitor tracing, and more from indistinguishability obfuscation. In *Crypto 2014, part I*, edited by J. A. Garay and R. Gennaro, pp. 480–499, Lecture Notes in Comput. Sci. 8616, Springer, Heidelberg, 2014.

[34] R. B. Boppana, J. Håstad, and S. Zachos, Does co-NP have short interactive proofs? *Inform. Process. Lett.* **25** (1987), no. 2, 127–132.

[35] E. Boyle, G. Couteau, N. Gilboa, and Y. Ishai, Compressing vector OLE. In *ACM CCS 2018*, edited by D. Lie, M. Mannan, M. Backes, and X. Wang, pp. 896–912, ACM Press, 2018.

[36] E. Boyle, G. Couteau, N. Gilboa, Y. Ishai, L. Kohl, and P. Scholl, Correlated pseudorandom functions from variable-density LPN In *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16–19, 2020*, pp. 1069–1080, IEEE, 2020.

[37] E. Boyle, G. Couteau, N. Gilboa, Y. Ishai, L. Kohl, P. Rindal, and P. Scholl, Efficient two-round OT extension and silent non-interactive secure computation. In *ACM CCS 2019*, edited by L. Cavallaro, J. Kinder, X. Wang, and J. Katz, pp. 291–308, ACM Press, 2019.

[38] Z. Brakerski, Y. T. Kalai, J. Katz, and V. Vaikuntanathan, Overcoming the hole in the bucket: public-key cryptography resilient to continual memory leakage. In *51st FOCS*, pp. 501–510, IEEE Computer Society Press, 2010.

[39] Z. Brakerski and G. N. Rothblum, Virtual black-box obfuscation for all circuits via generic graded encoding. In *Theory of Cryptography – 11th Theory of Cryptography Conference, TCC 2014, San Diego, CA, USA, February 24–26, 2014. Proceedings*, pp. 1–25, Lecture Notes in Comput. Sci. 8349, Springer, 2014.

[40] A. Broadbent and R. A. Kazmi, Indistinguishability obfuscation for quantum circuits of low t-count. *IACR Cryptol. ePrint Arch.* **2020** (2020), 639.
[41] R. Canetti, H. Lin, S. Tessaro, and V. Vaikuntanathan, Obfuscation of probabilistic circuits and applications. In *TCC 2015, part II*, edited by Y. Dodis and J. B. Nielsen, pp. 468–497, Lecture Notes in Comput. Sci. 9015, Springer, Heidelberg, 2015.
[42] J. H. Cheon, K. Han, C. Lee, H. Ryu, and D. Stehlé, Cryptanalysis of the multilinear map over the integers. In *Advances in Cryptology – EUROCRYPT 2015 – 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26–30, 2015, Proceedings, Part I*, pp. 3–12, Lecture Notes in Comput. Sci. 9056, Springer, 2015.
[43] J. H. Cheon, C. Lee, and H. Ryu, Cryptanalysis of the new clt multilinear maps. Cryptology ePrint Archive, Report 2015/934, 2015. http://eprint.iacr.org/.
[44] WhibOx Contest 2021—CHES 2021 Challenge. 2021, https://whibox.io/contests/2021/.
[45] J. Cook, O. Etesami, R. Miller, and L. Trevisan, Goldreich's one-way function candidate and myopic backtracking algorithms. In *TCC 2009*, edited by O. Reingold, pp. 521–538, Lecture Notes in Comput. Sci. 5444, Springer, Heidelberg, 2009.
[46] S. A. Cook, The complexity of theorem-proving procedures. In *Proceedings of the 3rd annual ACM symposium on theory of computing, May 3–5, 1971, Shaker Heights, Ohio, USA*, edited by M. A. Harrison, R. B. Banerji, and J. D. Ullman, pp. 151–158, ACM Press, 1971.
[47] J.-S. Coron, T. Lepoint, and M. Tibouchi, Practical multilinear maps over the integers. In *Crypto 2013, part I*, edited by R. Canetti and J. A. Garay, pp. 476–493, Lecture Notes in Comput. Sci. 8042, Springer, Heidelberg, 2013.
[48] G. Couteau, A. Dupin, P. Méaux, M. Rossi, and Y. Rotella, On the concrete security of Goldreich's pseudorandom generator. In *Asiacrypt 2018, part II*, edited by T. Peyrin and S. Galbraith, pp. 96–124, Lecture Notes in Comput. Sci. 11273, Springer, Heidelberg, 2018.
[49] M. Cryan and P. B. Miltersen, On pseudorandom generators in NC. In *MFCS 2001*, pp. 272–284, Lecture Notes in Comput. Sci. 2136, Springer, 2001.
[50] W. Diffie and M. E. Hellman, New directions in cryptography. *IEEE Trans. Inf. Theory* **22** (1976), no. 6, 644–654.
[51] N. Döttling, S. Garg, D. Gupta, P. Miao, and P. Mukherjee, Obfuscation from low noise multilinear maps. *IACR Cryptol. ePrint Arch.* **2016** (2016), 599.
[52] N. Döttling, S. Ghosh, J. B. Nielsen, T. Nilges, and R. Trifiletti, TinyOLE: efficient actively secure two-party computation from oblivious linear function evaluation. In *ACM CCS 2017*, edited by B. M. Thuraisingham, D. Evans, T. Malkin, and D. Xu, pp. 2263–2276, ACM Press, 2017.

[53] P. Farshim, J. Hesse, D. Hofheinz, and E. Larraia, Graded encoding schemes from obfuscation. In *PKC 2018, part II*, edited by M. Abdalla and R. Dahab, pp. 371–400, Lecture Notes in Comput. Sci. 10770, Springer, Heidelberg, 2018.

[54] S. Garg, C. Gentry, and S. Halevi, Candidate multilinear maps from ideal lattices. In *Eurocrypt 2013*, edited by T. Johansson and P. Q. Nguyen, pp. 1–17, Lecture Notes in Comput. Sci. 7881, Springer, Heidelberg, 2013.

[55] S. Garg, C. Gentry, S. Halevi, M. Raykova, A. Sahai, and B. Waters, Candidate indistinguishability obfuscation and functional encryption for all circuits. In *54th FOCS*, pp. 40–49, IEEE Computer Society Press, 2013.

[56] R. Gay, A. Jain, H. Lin, and A. Sahai, Indistinguishability obfuscation from simple-to-state hard problems: New assumptions, new techniques, and simplification. In *EUROCRYPT 2021*, pp. 97–126, Lecture Notes in Comput. Sci. 12698, Springer, 2021.

[57] C. Gentry, S. Gorbunov, and S. Halevi, Graph-induced multilinear maps from lattices. In *TCC 2015, part II*, edited by Y. Dodis and J. B. Nielsen, pp. 498–527, Lecture Notes in Comput. Sci. 9015, Springer, Heidelberg, 2015.

[58] C. Gentry, C. S. Jutla, and D. Kane, Obfuscation using tensor products. *Electron. Colloq. Comput. Complex.* **25** (2018), 149.

[59] S. Ghosh, J. B. Nielsen, and T. Nilges, Maliciously secure oblivious linear function evaluation with constant overhead. In *Asiacrypt 2017, part I*, edited by T. Takagi and T. Peyrin, pp. 629–659, Lecture Notes in Comput. Sci. 10624, Springer, Heidelberg, 2017.

[60] E. N. Gilbert, A comparison of signalling alphabets. *Bell Syst. Tech. J.* **31** (1952), no. 3, 504–522.

[61] O. Goldreich, Candidate one-way functions based on expander graphs. *Electron. Colloq. Comput. Complex.* **7** (2000), no. 90.

[62] J. Groth and A. Sahai, Efficient non-interactive proof systems for bilinear groups. In *Eurocrypt 2008*, edited by N. P. Smart, pp. 415–432, Lecture Notes in Comput. Sci. 4965, Springer, Heidelberg, 2008.

[63] Y. Hu and H. Jia, Cryptanalysis of GGH map. *IACR Cryptol. ePrint Arch.* **2015** (2015), 301.

[64] Y. Ishai and E. Kushilevitz, Perfect constant-round secure computation via perfect randomizing polynomials. In *ICALP*, pp. 244–256, Lecture Notes in Comput. Sci. 2380, Springer, 2002.

[65] Y. Ishai, E. Kushilevitz, R. Ostrovsky, and A. Sahai, Cryptography with constant computational overhead. In *40th ACM STOC*, edited by R. E. Ladner and C. Dwork, pp. 433–442, ACM Press, 2008.

[66] Y. Ishai, M. Prabhakaran, and A. Sahai, Founding cryptography on oblivious transfer—efficiently. In *Crypto 2008*, edited by D. Wagner, pp. 572–591, Lecture Notes in Comput. Sci. 5157, Springer, Heidelberg, 2008.

[67] A. Jain, H. Lin, C. Matt, and A. Sahai, How to leverage hardness of constant-degree expanding polynomials over $\mathbb{R}$ to build $i\mathcal{O}$. In *Eurocrypt 2019, part I*, edited by Y. Ishai and V. Rijmen, pp. 251–281, Lecture Notes in Comput. Sci. 11476, Springer, Heidelberg, 2019.
[68] A. Jain, H. Lin, and A. Sahai, Indistinguishability obfuscation from well-founded assumptions. In *STOC'21: 53rd annual ACM SIGACT symposium on theory of computing, virtual event, Italy, June 21–25, 2021*, edited by S. Khuller and V. V. Williams, pp. 60–73, ACM Press, 2021.
[69] A. Jain, H. Lin, and A. Sahai, Indistinguishability obfuscation without lattices. Unpublished manuscript, 2021.
[70] D. Khurana, V. Rao, and A. Sahai, Multi-party key exchange for unbounded parties from indistinguishability obfuscation. In *Asiacrypt 2015, part I*, edited by T. Iwata and J. H. Cheon, pp. 52–75, Lecture Notes in Comput. Sci. 9452, Springer, Heidelberg, 2015.
[71] I. Komargodski, T. Moran, M. Naor, R. Pass, A. Rosen, and E. Yogev, One-way functions and (im)perfect obfuscation. In *55th FOCS*, pp. 374–383, IEEE Computer Society Press, 2014.
[72] I. Komargodski, M. Naor, and E. Yogev, Secret-sharing for NP. In *Asiacrypt 2014, part II*, edited by P. Sarkar and T. Iwata, pp. 254–273, Lecture Notes in Comput. Sci. 8874, Springer, Heidelberg, 2014.
[73] P. K. Kothari, R. Mori, R. O'Donnell, and D. Witmer, Sum of squares lower bounds for refuting any CSP. In *49th ACM STOC*, edited by H. Hatami, P. McKenzie, and V. King, pp. 132–145, ACM Press, 2017.
[74] H. Lin, Indistinguishability obfuscation from constant-degree graded encoding schemes. In *Eurocrypt 2016, part I*, edited by M. Fischlin and J.-S. Coron, pp. 28–57, Lecture Notes in Comput. Sci. 9665, Springer, Heidelberg, 2016.
[75] H. Lin, Indistinguishability obfuscation from SXDH on 5-linear maps and locality-5 PRGs. In *Crypto 2017, part I*, edited by J. Katz and H. Shacham, pp. 599–629, Lecture Notes in Comput. Sci. 10401, Springer, Heidelberg, 2017.
[76] H. Lin and C. Matt, Pseudo flawed-smudging generators and their application to indistinguishability obfuscation. *IACR Cryptol. ePrint Arch.* **2018** (2018), 646.
[77] H. Lin, R. Pass, K. Seth, and S. Telang, Indistinguishability obfuscation with non-trivial efficiency. In *PKC 2016, part II*, edited by C.-M. Cheng, K.-M. Chung, G. Persiano, and B.-Y. Yang, pp. 447–462, Lecture Notes in Comput. Sci. 9615, Springer, Heidelberg, 2016.
[78] H. Lin, R. Pass, K. Seth, and S. Telang, Output-compressing randomized encodings and applications. In *TCC 2016-a, part I*, edited by E. Kushilevitz and T. Malkin, pp. 96–124, Lecture Notes in Comput. Sci. 9562, Springer, Heidelberg, 2016.
[79] H. Lin and S. Tessaro, Indistinguishability obfuscation from trilinear maps and block-wise local PRGs. In *Crypto 2017, part I*, edited by J. Katz and H. Shacham, pp. 630–660, Lecture Notes in Comput. Sci. 10401, Springer, Heidelberg, 2017.

[80] H. Lin and V. Vaikuntanathan, Indistinguishability obfuscation from DDH-like assumptions on constant-degree graded encodings. In *57th FOCS*, edited by I. Dinur, pp. 11–20, IEEE Computer Society Press, 2016.

[81] A. Lombardi and V. Vaikuntanathan, Limits on the locality of pseudorandom generators and applications to indistinguishability obfuscation. In *TCC 2017, part I*, edited by Y. Kalai and L. Reyzin, pp. 119–137, Lecture Notes in Comput. Sci. 10677, Springer, Heidelberg, 2017.

[82] D. Micciancio and P. Mol, Pseudorandom knapsacks and the sample complexity of LWE search-to-decision reductions. In *Crypto 2011*, edited by P. Rogaway, pp. 465–484, Lecture Notes in Comput. Sci. 6841, Springer, Heidelberg, 2011.

[83] D. Micciancio and C. Peikert, Hardness of SIS and LWE with small parameters. In *Crypto 2013, part I*, edited by R. Canetti and J. A. Garay, pp. 21–39, Lecture Notes in Comput. Sci. 8042, Springer, Heidelberg, 2013.

[84] E. Miles, A. Sahai, and M. Zhandry, Annihilation attacks for multilinear maps: Cryptanalysis of indistinguishability obfuscation over GGH13. In *Advances in cryptology—CRYPTO*, 2016.

[85] B. Minaud and P.-A. Fouque, Cryptanalysis of the new multilinear map over the integers. Cryptology ePrint Archive, Report 2015/941, 2015. http://eprint.iacr.org/.

[86] E. Mossel, A. Shpilka, and L. Trevisan, On e-biased generators in NC^0. In *44th FOCS*, pp. 136–145, IEEE Computer Society Press, 2003.

[87] R. O'Donnell and D. 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*, pp. 1–12, IEEE Computer Society, 2014.

[88] A. O'Neill, Definitional issues in functional encryption. *IACR Cryptol. ePrint Arch.* **2010** (2010), 556.

[89] T. Okamoto and K. Takashima, Fully secure functional encryption with general relations from the decisional linear assumption. In *Crypto 2010*, edited by T. Rabin, pp. 191–208, Lecture Notes in Comput. Sci. 6223, Springer, Heidelberg, 2010.

[90] R. Pass, K. Seth, and S. Telang, Indistinguishability obfuscation from semantically-secure multilinear encodings. In *Advances in Cryptology – CRYPTO 2014 – 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17–21, 2014, Proceedings, Part I*, pp. 500–517, Lecture Notes in Comput. Sci. 8616, Springer, 2014.

[91] O. Regev, On lattices, learning with errors, random linear codes, and cryptography. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing, Baltimore, MD, USA, May 22–24, 2005*, pp. 84–93, ACM, 2005.

[92] A. Sahai and B. Waters, How to use indistinguishability obfuscation: deniable encryption, and more. In *46th ACM STOC*, edited by D. B. Shmoys, pp. 475–484, ACM Press, 2014.

[93] A. Sahai and B. R. Waters, Fuzzy identity-based encryption. In *Eurocrypt 2005*, edited by R. Cramer, pp. 457–473, Lecture Notes in Comput. Sci. 3494, Springer, Heidelberg, 2005.

[94] V. Vaikunthananthan, Cs 294 lecture 4: worst-case to average-case reductions for LWE. Class at UC Berkeley, 2020, http://people.csail.mit.edu/vinodv/CS294/lecture4.pdf.

[95] R. Varshamov, Estimate of the number of signals in error correcting codes. *Dokl. Akad. Nauk SSSR* **117** (1957), 739–741.

[96] H. Wee, Functional encryption for quadratic functions from k-LIN, revisited. In *Theory of Cryptography – 18th International Conference, TCC 2020, Durham, NC, USA, November 16–19, 2020, Proceedings, Part I*, pp. 210–228, Lecture Notes in Comput. Sci. 12550, Springer, 2020.

[97] T. Yamakawa, S. Yamada, G. Hanaoka, and N. Kunihiro, Self-bilinear map on unknown order groups from indistinguishability obfuscation and its applications. In *Crypto 2014, part II*, edited by J. A. Garay and R. Gennaro, pp. 90–107, Lecture Notes in Comput. Sci. 8617, Springer, Heidelberg, 2014.

[98] A. C.-C. Yao, How to generate and exchange secrets (extended abstract). In *27th Annual Symposium on Foundations of Computer Science, Toronto, Canada, 27–29 October 1986*, pp. 162–167, IEEE Computer Society, 1986.

AAYUSH JAIN

UCLA, Los Angeles, CA, USA; and NTT Research, Sunnyvale, CA, USA, aayushjain1728@gmail.com

HUIJIA LIN

University of Washington, Seattle, WA, USA, rachel@cs.washington.edu

AMIT SAHAI

UCLA, Los Angeles, CA, USA, sahai@cs.ucla.edu