Indistinguishability Obfuscation from Well-Founded Assumptions

Aayush Jain
aayushjain1728@gmail.com
University of California
Los Angeles, CA, USA

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

Amit Sahai
sahai@cs.ucla.edu
University of California
Los Angeles, CA, USA

ABSTRACT

Indistinguishability obfuscation, introduced by [Barak et. al. Crypto 2001], aims to compile programs into unintelligible ones while preserving functionality. It is a fascinating and powerful object that has been shown to enable a host of new cryptographic goals and beyond. However, constructions of indistinguishability obfuscation have remained elusive, with all other proposals relying on heuristics or newly conjectured hardness assumptions.

In this work, we show how to construct indistinguishability obfuscation from subexponential hardness of four well-founded assumptions. We prove:

Informal Theorem: *Let $\tau \in (0,\infty), \delta \in (0,1), \epsilon \in (0,1)$ be arbitrary constants. Assume sub-exponential security of the following assumptions:*
- the Learning With Errors (LWE) assumption with subexponential modulus-to-noise ratio 2^{k^ϵ} and noises of magnitude polynomial in k, where k is the dimension of the LWE secret,
- the Learning Parity with Noise (LPN) assumption over general prime fields $\mathbb{Z}_p$ with polynomially many LPN samples and error rate $1/\ell^\delta$, where ℓ is the dimension of the LPN secret,
- the existence of a Boolean Pseudo-Random Generator (PRG) in NC^0 with stretch $n^{1+\tau}$, where n is the length of the PRG seed,
- the Decision Linear (DLIN) assumption on symmetric bilinear groups of prime order.
Then, (subexponentially secure) indistinguishability obfuscation for all polynomial-size circuits exists. Further, assuming only polynomial security of the aforementioned assumptions, there exists collusion resistant public-key functional encryption for all polynomial-size circuits.

CCS CONCEPTS

• **Theory of computation → Cryptographic primitives**.

KEYWORDS

indistinguishability obfuscation

STOC '21, June 21–25, 2021, Virtual, Italy

ACM ISBN 978-1-4503-8053-9/21/06…$15.00
https://doi.org/10.1145/3406325.3451093

ACM Reference Format:
Aayush Jain, Huijia Lin, and Amit Sahai. 2021. Indistinguishability Obfuscation from Well-Founded Assumptions. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing (STOC '21), June 21–25, 2021, Virtual, Italy.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3406325.3451093

1 INTRODUCTION

In this work, we study the notion of indistinguishability obfuscation ($i\mathcal{O}$) for general polynomial-size circuits [19, 66, 77]. $i\mathcal{O}$ requires that for any two circuits C_0 and C_1 of the same size, such that $C_0(x) = C_1(x)$ for all inputs x, we have that $i\mathcal{O}(C_0)$ is computationally indistinguishable to $i\mathcal{O}(C_1)$. Furthermore, the obfuscator $i\mathcal{O}$ should be computable in probabilistic polynomial time. The notion of $i\mathcal{O}$ has proven to be very powerful, with over a hundred papers published utilizing $i\mathcal{O}$ to enable a remarkable variety of applications in cryptography and complexity theory; indeed $i\mathcal{O}$ has even expanded the scope of cryptography (see, e.g. [17, 26, 44, 57, 58, 66, 67, 76, 83, 83, 84, 99, 111]).

Despite this success, until this work, all previously known $i\mathcal{O}$ constructions (see [68] and the references therein) required new hardness assumptions that were postulated specifically for showing security of the $i\mathcal{O}$ schemes proposed. Indeed, the process of understanding these assumptions has been tortuous, with several of these assumptions broken by clever cryptanalysis [18, 21, 33, 40, 53, 54, 60, 82, 85, 103, 105, 106]. The remaining standing ones are based on new and novel computational problems that are different in nature from well-studied computational problems (for instance, LWE with leakage on noises).

As a result, there has been a lack of clarity about the state of $i\mathcal{O}$ security [24]. Our work aims to place $i\mathcal{O}$ on *terra firma.*

Our contribution. We show how to construct $i\mathcal{O}$ from subexponential hardness of four well-founded assumptions. We prove:

Theorem 1.1. *(Informal) Let τ be arbitrary constants greater than 0, and δ, ϵ in $(0,1)$. Assume sub-exponential security of the following assumptions, where λ is the security parameter, p is a λ-bit prime, and the parameters ℓ, k, n below are large enough polynomials in λ:*

- *the LWE assumption over $\mathbb{Z}_p$ with subexponential modulus-to-noise ratio 2^{k^ϵ} and noises of magnitude polynomial in k, where k is the dimension of the LWE secret,*
- *the LPN assumption over general prime fields $\mathbb{Z}_p$ with polynomially many LPN samples and error rate $1/\ell^\delta$, where ℓ is the dimension of the LPN secret,*
- *the existence of a Boolean PRG in NC^0 with stretch $n^{1+\tau}$,*
- *the DLIN assumption on symmetric bilinear groups of prime order.*

Then, (subexponentially secure) indistinguishability obfuscation for all polynomial-size circuits exists.

All four assumptions 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 *iO*.

1.1 Assumptions in More Detail

We now describe each of these assumptions in more detail and briefly survey their history. Throughout the paper, when we say that a primitive or assumption is polynomially or subexponentially hard (alternatively, secure or indistinguishable), we mean that the hardness of the primitive or assumption holds against all adversaries that run in *polynomial* time in the security parameter λ with at most negligible or subexponential advantage $2^{-\lambda^\epsilon}$ for some $\epsilon > 0$ respectively. More precisely, the security level of the assumptions below is first measured in their own parameters, namely, the modulus length $\lceil \log p \rceil$ of pairing groups for DLIN over symmetric bilinear groups, the dimension k of the LWE secret, the dimension ℓ of the LPN secret, and the seed length n of the PRG in NC_0. In our construction, modulus length $\lceil \log p \rceil$ is set to λ, and k, ℓ, n are sufficiently large polynomials in λ. Therefore, polynomial or subexponential hardness of these assumptions measured in their own parameters translates into polynomial or subexponential hardness in λ.

The DLIN *Assumption:* The Decisional Linear assumption (DLIN) is stated as follows: For an appropriate λ-bit prime p, two groups $\mathbb{G}$ and $\mathbb{G}_T$ are chosen of order p 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. Then, the DLIN assumption requires that the following computational indistinguishability holds:

$$\left\{(g^x, g^y, g^{xr}, g^{ys}, g^{r+s}) \;\middle|\; x, y, r, s \leftarrow \mathbb{Z}_p\right\}$$
$$\approx_c \left\{(g^x, g^y, g^{xr}, g^{ys}, g^{z}) \;\middle|\; x, y, r, s, z \leftarrow \mathbb{Z}_p\right\}$$

This assumption was first introduced in the 2004 work of Boneh, Boyen, and Shacham [30]. 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, Non-Interactive Zero Knowledge, etc. (See, e.g. [23, 42, 81, 93, 109]).

The LWE *Assumption:* The Learning With Errors LWE assumption with respect to a modulus q, dimension k, sample complexity n, and discrete Gaussian distribution χ over integers states that the following computational indistinguishability holds:

$$\{\mathbf{A}, \boldsymbol{s} \cdot \mathbf{A} + \boldsymbol{e} \bmod q \;|\; \mathbf{A} \leftarrow \mathbb{Z}_q^{k \times n},\ \boldsymbol{s} \leftarrow \mathbb{Z}_q^{1 \times k},\ \boldsymbol{e} \leftarrow \chi^{1 \times n}\}$$
$$\approx_c \{\mathbf{A}, \boldsymbol{u} \;|\; \mathbf{A} \leftarrow \mathbb{Z}_q^{k \times n},\ \boldsymbol{u} \leftarrow \mathbb{Z}_q^{1 \times n}\}$$

In this work, the dimension of the secret k is a sufficiently large polynomial in λ. Moreover, the sample complexity is polynomial $n(k)$, and we require the modulus-to-noise ratio to be subexponential in k, i.e., 2^{k^ϵ} for an arbitrary $\epsilon > 0$, and the magnitude of the noises to be polynomial in k.

This assumption was first stated in the work of [110]. LWE has been used extensively to construct applications such as Leveled Fully Homomorphic Encryption [41, 43, 71], Lockable Obfuscation [80, 114], Attribute Based Encryption [32, 78, 79] and Universal Thresholdizers [31].

The existence of PRG*s in* NC^0*:* The assumption of the existence of a Boolean Pseudo-Random Generator PRG in NC^0 states that there exists a Boolean function $G : \{0,1\}^n \to \{0,1\}^m$ 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:

$$\{G(\boldsymbol{\sigma}) \;|\; \boldsymbol{\sigma} \leftarrow \{0,1\}^n\} \approx_c \{\boldsymbol{y} \;|\; \boldsymbol{y} \leftarrow \{0,1\}^m\}$$

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 [62, 75, 87] 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 [9, 10, 12, 13, 15, 16, 28, 29, 59, 61, 62, 75, 101, 107, 108].

LPN *over large fields:* Like LWE, the Learning Parity with Noise LPN assumption over finite fields $\mathbb{Z}_p$ is also 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:

$$\{\mathbf{A}, \boldsymbol{s} \cdot \mathbf{A} + \boldsymbol{e} \bmod p \;|\; \mathbf{A} \leftarrow \mathbb{Z}_p^{\ell \times n},\ \boldsymbol{s} \leftarrow \mathbb{Z}_p^{1 \times \ell},\ \boldsymbol{e} \leftarrow \mathcal{D}_r^{1 \times n}\}$$
$$\approx_c \{\mathbf{A}, \boldsymbol{u} \;|\; \mathbf{A} \leftarrow \mathbb{Z}_p^{\ell \times n},\ \boldsymbol{u} \leftarrow \mathbb{Z}_p^{1 \times n}\}.$$

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 $1 - 1/\ell^\delta$. Thus, the difference between LWE and LPN is the structure of the error distribution. In LWE the error vector is a random (polynomially) bounded vector. In LPN, it is a random sparse vector, but where it is nonzero, the entries have large expectation. Again, 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 [74] and Varshamov [112] showed that random linear codes possessed remarkably strong minimum distance properties. However, since then, almost no 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 [27], and formally defined for general finite fields and parameters in 2009 [88], under the name "Assumption 2".

While in [88], the assumption was used when the error rate is 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 $\mathbb{F}_2$, and used to build public-key encryption over $\mathbb{F}_p$ 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 $\delta > 0$ was explicitly posed by [35], in the context of building efficient secure two-party and multi-party protocols for arithmetic computations.

Recently, the LPN assumption has led to a wide variety of applications (see for example, [11, 14, 22, 35, 36, 63, 73, 88]). A comprehensive review of known attacks on LPN over large fields, for the parameter settings we are interested in, was given in [35, 37]. 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 $n(\ell)$.

The works of [10, 15] 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 i'th output bit of $f_{G,Q}$ is computed by evaluating Q on the d input bits indexed by nodes in the i'th hyperedge. The one-wayness of $\mathcal{F}_{Q,m}$ for proper choices of Q, m has been conjectured and studied in [12, 28, 59, 75, 107]. The works of [10, 15] 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 noticable probability by adversaries of some subexponential size.

1.2 Our Ideas in a Nutshell

Previous works [6, 7, 89, 90, 102] led to the recent construction of $i\mathcal{O}$ in [68] based on LWE, DLIN over symmetric bilinear groups, and one other object, that we will encapsulate as a *structured-seed* PRG (sPRG) with polynomial stretch and special efficiency properties[1]. In an sPRG, the seed consists of both a public and private part. The pseudorandomness property of the sPRG should hold even when the adversary can see the public seed in addition to the output of the sPRG. Crucially, the output of the sPRG should be computable by a *degree-2* computation in the private seed (where the coefficients of this degree-2 computation are obtained through constant-degree computations on the public seed).

Our key innovation is a simple way to leverage LPN over fields to build an sPRG. The starting point for our construction is the following observation. Assuming LPN and that G is an (ordinary) PRG in NC^0 with stretch $m(n)$, we immediately have the following computational indistinguishability:

$$\{(\boldsymbol{A},\ \boldsymbol{b} = \boldsymbol{s}\cdot\boldsymbol{A} + \boldsymbol{e} + \boldsymbol{\sigma},\ G(\boldsymbol{\sigma}))\} \approx_c \{(\boldsymbol{A},\ \boldsymbol{u},\ \boldsymbol{w})\}$$

where $\boldsymbol{A} \leftarrow \mathbb{Z}_p^{\ell\times n}$, $\boldsymbol{s} \leftarrow \mathbb{Z}_p^{1\times\ell}$, $\boldsymbol{e} \leftarrow \mathcal{D}_r^{1\times n}(p)$, $\boldsymbol{\sigma} \leftarrow \{0,1\}^{1\times n}$, $\boldsymbol{u} \leftarrow \mathbb{Z}_p^{1\times n}$, and $\boldsymbol{w} \leftarrow \{0,1\}^{1\times m(n)}$.

Roughly speaking, we can think of both $\boldsymbol{A}$ and $\boldsymbol{b}$ above as being public. All that remains is to show that the computation of $G(\boldsymbol{\sigma})$ can be performed using a degree-2 computation in a short-enough specially-prepared secret seed. Because G is an arbitrary PRG in NC^0, it will not in general be computable by a degree-2 polynomial in $\boldsymbol{\sigma}$. To accomplish this goal, we crucially leverage the *sparseness* of the LPN error $\boldsymbol{e}$. The evaluation of our sPRG can be viewed to take two steps: First, homomorphically evaluate the PRG on an LPN-encryption of the seed $\boldsymbol{\sigma}$ (i.e., $\mathbf{A}, \boldsymbol{b}$ above) to obtain an LPN-encryption of the PRG output $G(\boldsymbol{\sigma})$ that is however corrupted with *sparse* errors. Next, decrypt and correct the sparse errors all in just degree 2, by means of a simple pre-computation idea. In particular, the precomputation compresses the *sparse* errors to be corrected into vectors of sublinear length that later can be expanded back using a degree 2 computation. A gentle overview is provided in Section 4, followed by our detailed construction and analysis.

1.3 Implications of $i\mathcal{O}$

The notion of $i\mathcal{O}$ occupies an intriguing and influential position in complexity theory and cryptography. Interestingly, if NP $\subseteq$ BPP, then $i\mathcal{O}$ exists for the class of all polynomial-size circuits, because if NP $\subseteq$ 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 NP $\not\subseteq$ io-BPP, then in fact the existence of $i\mathcal{O}$ for polynomial-size circuits implies that one-way functions exist [96]. And a large body of work has shown that $i\mathcal{O}$ plus one-way functions imply a vast array of cryptographic objects, so much so that $i\mathcal{O}$ has been conjectured to be a "central hub" [96, 111] 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 subset of these implications as corollaries.

Corollary 1.2 (Informal). *Assume the* subexponential *hardness of the four assumptions in Theorem 1.1, we have:*

- *Multiparty non-interactive key exchange in the plain model (without trusted setup), e.g., [34, 95].*
- *Adaptively secure succinct garbled RAM, where the size of the garbled program is* $\mathsf{poly}(\lambda, \log T)|P|$ *depending linearly on the description size of the RAM program P, the size of the garbled input is* $\mathsf{poly}(\lambda)|x|$ *depending linearly on the size of the input x, and evaluation time is quasilinear in the running time of P on x [5, 8, 25, 46, 48, 49, 52, 100].*
- *Indistinguishability obfuscation for RAM, where the size of obfuscated program is* $\mathsf{poly}(\lambda, n, |P|)$ *where $|P|$ is the description size of the RAM program P and n is its input length [5, 8, 25, 46, 48, 49, 52, 100].*
- *Selectively sound and perfectly zero-knowledge Succinct Non-interactive ARGument (SNARG) for any NP language with statements up to a bounded polynomial size in the CRS model,*

[1] In early version of [68], the notion of sPRG was implicit in the sense that [68] directly proposed a candidate sPRG based on a new assumption without formalizing the notion of sPRG. They then constructed $i\mathcal{O}$ using this candidate sPRG, LWE, and DLIN. After this work formalized the notion of sPRG, [68] has updated their construction to replace their specific candidate with a general sPRG.

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* $\mathsf{poly}(\lambda)$ *[111]*[2].
- *Sender deniable encryption [111], and fully deniable interactive encryption [51].*
- *Constant round concurrent zero-knowledge protocols for any NP language [57].*
- *(Symmetric or asymmetric) multilinear maps with boudned polynomial multilinear degrees, following [2, 3, 65], and self-bilinear map over composite and unknown order group, assuming additionally the polynomial hardness of factoring [115].*
- *Correlation intractable functions for all sparse relations verifiable in bounded polynoimal size, assuming additionally the polynomial hardness of input hiding obfuscators for evasive circuits [47], or for all sparse relations, assuming additionally the exponential optimal hardness of input hiding obfuscators for multibit point functions [94].*
- *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 [97].*

Corollary 1.3 (Informal). *Assume the* polynomial *hardness of the four assumptions in Theorem 1.1, we have:*

- *Attribute Based Encryption (ABE) for unbounded-depth polynomial-size circuits, following as a special case of functional encryption for unbounded-depth polynomial size circuits.*
- *Fully homomorphic encryption scheme for unbounded-depth polynomial size circuits (without relying on circular security), assuming slighly superpolynomial hardness of the four assumptions [50].*
- PPAD *hardness [1, 26, 67, 86, 98].*

Regarding PPAD hardness, another line of beautiful works [45, 55, 56, 64, 92, 104] showed that the hardness of #SAT reduces to that of PPAD, assuming the adaptive soundness of applying Fiat-Shamir to certain protocols. Most recently, this led to basing the PPAD hardness on that of #SAT and the sub-exponential LWE assumption [92]. In comparison, relying on [67], using our Functional Encryption construction, we can base the PPAD hardness on the polynomial security of the four assumptions we make.

1.4 Other Recent Works

We briefly discuss three recent works [38, 69, 113][3] that develop a completely independent line of attack for constructing $i\mathcal{O}$. At present this line still requires new hardness assumptions, but has other potential advantages, as we explain below.

Gay and Pass [69] proved the security of a variant of the candidate $i\mathcal{O}$ of [39], based on the subexponential hardness of *i)* LWE with subexponential modulus-to-noise ratio, and *ii)* a new circular security leakage resilience conjecture, referred to as *1-circular Shielded Randomness Leakage (SRL) resilient security*, on the Gentry, Sahai, and Waters (GSW) homomorphic encryption scheme [71].

The work of [38] proved the security of a similar construction based on a new variant of the 2-circular SRL security assumption proposed by [69], with respect to GSW and a packed variant of the dual-Regev encryption scheme [70], which requires SRL security to hold in the presence of a length-2 chain of circular encryptions of the secret keys of GSW and packed dual-Regev. They also suggest another assumption that may be a potential relaxation that considers a key-randomness encryption cycle, instead of a key cycle.

Wichs and Wee [113] proposed a different construction of $i\mathcal{O}$ based on a new encryption scheme called dual-GSW that "marries" together dual-Regev and GSW encryption schemes. Their new assumption also has a circular security flavor, but is significantly different from the circular SRL security that [38, 69] proposed.

Comparing with the above constructions, our construction has the advantage in relying solely on well-founded assumptions that have a long history of study. On the other hand, these constructions are based only on lattices and have the advantage of being plausibly quantum-secure, whereas the SXDH assumption on bilinear maps that our construction relies on is known to be broken by polynomial-time quantum algorithms.

2 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 $\{1, .., n\}$. A function $\mathsf{negl} : \mathbb{N} \to \mathbb{R}$ is negligible 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 security parameter, we mean constant degree polynomials that take positive value on non negative inputs. We denote by $\mathsf{poly}(\lambda)$ an arbitrary polynomial in λ satisfying the above requirements of non-negativity. We denote vectors by bold-faced letters such as $\boldsymbol{b}$ and $\boldsymbol{u}$. Matrices will be denoted by capitalized bold-faced letters for such as $\boldsymbol{A}$ and $\boldsymbol{M}$. For any $k \in \mathbb{N}$, we denote by $\boldsymbol{v}^{\otimes k}$ the vector obtained by tensoring $\boldsymbol{v}$ with itself for k times. This vector contains all the monomials in the variables inside $\boldsymbol{v}$ of degree exactly k.

We also introduce two new notations. First, for any vector $\boldsymbol{v}$ we refer by $\mathsf{dim}(\boldsymbol{v})$ the dimension of vector $\boldsymbol{v}$. For any matrix $\mathbf{M} \in \mathbb{Z}_q^{n_1 \times n_2}$, we denote by $|\mathbf{M}|$ the bit length of $\mathbf{M}$. In this case, $|\mathbf{M}| = n_1 \cdot n_2 \cdot \log_2 q$. We also overload this operator in that, for any set S, we use $|S|$ to denote the cardinality of S. The meaning should be inferred from context.

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 exists an $\epsilon \in (0, 1)$ and a constant $c > 0$ such that $a < b^{1-\epsilon} \cdot \lambda^c$ for all large enough $\lambda, n \in \mathbb{N}$. 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 $f \in \mathbb{Z}[\boldsymbol{x} = (x_1, \ldots, x_n)]$, mapping $\{0,1\}^n$ to $\{0,1\}$. At times, we consider a mapping of such polynomial $f \in \mathbb{Z}[\boldsymbol{x}]$ into a polynomial g over $\mathbb{Z}_p[\boldsymbol{x}]$ for some prime p. This is simply obtained by reducing the coefficients of f modulo p and then evaluating the polynomial

[2] This construction does not contradict the lower bound result by [72] showing that it is impossible to base the adaptive soundness of SNARGs on falsifiable assumptions via black-box reductions, since this construction only achieves selective soundness.
[3] These works were posted online very shortly after the initial posting of our work.

over $\mathbb{Z}_p$. Observe that $g(\boldsymbol{x}) = f(\boldsymbol{x}) \mod p$ for every $\boldsymbol{x} \in \{0,1\}^n$ as $f(\boldsymbol{x}) \in \{0,1\}$ for every such $\boldsymbol{x}$. Furthermore, given any NC^0 function F, finding these representations take polynomial time.

Definition 2.1 (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 $\epsilon : \mathbb{N} \to [0,1]$ if for every probabilistic polynomial time adversary $\mathcal{A}$ it holds that: For every sufficiently large $\lambda \in \mathbb{N}$,

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

We say that two ensembles are polynomially indistinguishable if they are ϵ-indistinguishable for $\epsilon(\lambda) = \mathsf{negl}(\lambda)$ for some negligible negl, and that two ensembles are sub-exponentially indistinguishable if they are ϵ-indistinguishable for $\epsilon(\lambda) = 2^{-\lambda^c}$ for some positive real number c.

Below if the security a primitive or the hardness of an assumption are defined through indistinguishability, we say the primitive or assumption is ϵ secure, hard, or indistinguishable, or (subexponentially) secure, hard, or indistinguishable if the appropriate ϵ-indistinguishability or (subexponentially) indistinguishability holds.

Remark 2.1 (On security levels). *We denote by λ the global security parameter and set all other parameters as functions of λ. The security of different assumptions,* DLIN *over symmetric bilinear groups,* LWE, LPN, *and* PRG *in* NC^0 *depends on their own parameters, namely the order p of the bilinear pairing group, the dimension of the* LWE *secrets, the dimension of the* LPN *secrets, and the length of the* PRG *seeds, and thus are indirectly related to the global security parameter λ. For our FE construction, we need all assumptions to be polynomially hard (in λ), and for our $i\mathcal{O}$ construction, we need all assumptions to be sub-exponentially hard (in λ).*

Indistinguishability Obfuscation. We now define our object of interest, Indistinguishability Obfuscation ($i\mathcal{O}$). The notion of indistinguishability obfuscation (iO), first conceived by Barak et al. [20], guarantees that the obfuscation of two circuits are computationally indistinguishable as long as they both are equivalent circuits, i.e., the output of both the circuits are the same on every input. Formally,

Definition 2.2 (Indistinguishability Obfuscator (iO) for Circuits). A uniform PPT algorithm $i\mathcal{O}$ is called a γ-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, every input $x \in \{0,1\}^n$, we have that

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

- **Indistinguishability:** For every two ensembles $\{C_{0,\lambda}\}$ and $\{C_{1,\lambda}\}$ of polynomial-sized circuits that have the same size, input length, and output length, and are functionally equivalent, that is, $\forall\lambda$, $C_{0,\lambda}(x) = C_{1,\lambda}(x)$ for every input x, the following distributions are polynomially indistinguishable.

$$\{i\mathcal{O}(1^\lambda, C_{0,\lambda})\} \qquad \{i\mathcal{O}(1^\lambda, C_{1,\lambda})\}$$

LPN over Fields Assumption. In this work, we use the LPN assumption over a large field. This assumption has been used in a various works (see for example, [11, 14, 22, 35, 36, 63, 73, 88]). We adopt the following definition from [35]. We use the assumption with error rate $\ell^{-\delta}$, for any constant $\delta > 0$, where ℓ is the dimension of the LPN secret.

We set up some notation for the definition below. Let p be any prime modulus. We define the distribution $\mathcal{D}_r(p)$ as the distribution that outputs 0 with probability $1 - r$ and a random element from $\mathbb{Z}_p$ with the remaining probability. Below we define δ-$\mathsf{LPN}_\mathbb{F}$ Assumption [11, 27, 35, 88].

Definition 2.3 (δ-$\mathsf{LPN}_\mathbb{F}$ Assumption). Let λ be the security parameter. We denote positive integer dimension $\ell = \ell(\lambda)$, noise rate $r = r(\ell) \in (0,1]$, and positive integer sample complexity $n = n(\ell)$.

We say that the δ-$\mathsf{LPN}_\mathbb{F}$ Assumption is γ-hard if the following holds:

For any constant $\eta_p > 0$, any function $p : \mathbb{N} \to \mathbb{N}$ s.t., for every $\lambda \in \mathbb{N}$, $p(\lambda)$ is a prime of k^{η_p} bits, any constants $\eta_\ell \geq 1$, and $\eta_n > 0$, we set $p = p(\lambda)$, $\ell = \ell(\lambda) = \lambda^{\eta_\ell}$, $n = n(\ell) = \ell^{\eta_n}$, and $r = r(\ell) = \ell^{-\delta}$, and we require that the following two distributions are γ-indistinguishable:

$$\left\{ (\boldsymbol{A}, \boldsymbol{b} = \boldsymbol{s}\cdot\boldsymbol{A} + \boldsymbol{e}) \ | \ \boldsymbol{A} \leftarrow \mathbb{Z}_p^{\ell\times n}, \ \boldsymbol{s} \leftarrow \mathbb{Z}_p^{1\times\ell}, \ \boldsymbol{e} \leftarrow \mathcal{D}_r^{1\times n}(p) \right\}$$

$$\left\{ (\boldsymbol{A}, \boldsymbol{u}) \ | \ \boldsymbol{A} \leftarrow \mathbb{Z}_p^{\ell\times n}, \ \boldsymbol{u} \leftarrow \mathbb{Z}_p^{1\times n} \right\}$$

We refer the reader to [35, 37] for a comprehensive discussion of the history and security of this assumption.

3 DEFINITION OF STRUCTURED-SEED PRG

Definition 3.1 (Syntax of Structured-Seed Pseudo-Random Generators (sPRG)). Let τ be a positive constant. A structured-seed Boolean PRG, sPRG, with stretch τ that maps $(n\cdot\mathsf{poly}(\lambda))$-bit binary strings into $(m = n^\tau)$-bit strings, where poly is a fixed polynomial, is defined by the following PPT algorithms:

- $\mathsf{IdSamp}(1^\lambda, 1^n)$ samples a function index I.
- $\mathsf{SdSamp}(I)$ jointly samples two binary strings, a public seed and a private seed, $\mathsf{sd} = (P, S)$. The combined length of these strings is $n \cdot \mathsf{poly}(\lambda)$.
- $\mathsf{Eval}(I, \mathsf{sd})$ computes a string in $\{0,1\}^m$.

Remark 3.1 (Polynomial Stretch.). *We say that an* sPRG *has polynomial stretch if $\tau > 1$ for some constant τ.*

Remark 3.2 (On $\mathsf{poly}(\lambda)$ multiplicative factor in the seed length.). *As opposed to a standard Boolean* PRG *definition where the length of the output is set to be n^τ where n is the seed length, we allow the length of the seed to increase multiplicatively by a fixed polynomial* poly *in a parameter λ. Looking ahead, one should view n as an arbitrary large polynomial in λ, and hence* sPRG *will be expanding in length.*

Definition 3.2 (Security of sPRG). A structured-seed Boolean PRG, sPRG, satisfies

γ-**pseudorandomness:** For every integer $n = n(\lambda) \geq \lambda$, the following distributions are γ indistinguishable.

$$\{I, \ P, \ \mathsf{Eval}(I, P)\}$$

$$\{I, \ P, \ \boldsymbol{r}\},$$

where $I \leftarrow \mathsf{IdSamp}(1^\lambda, 1^n)$, $\mathsf{sd} \leftarrow \mathsf{SdSamp}(I)$ and $\boldsymbol{r} \leftarrow \{0,1\}^{m(n)}$.

Definition 3.3 (Complexity and degree of sPRG*).* Let $d \in \mathbb{N}$, let $\lambda \in \mathbb{N}$ and $n = n(\lambda)$ be arbitrary positive polynomial in λ, and $p = p(\lambda)$ denote a prime modulus which is an efficiently computable function in λ. Let $\mathbb{C}$ be a complexity class. A sPRG has complexity $\mathbb{C}$ in the public seed and degree d in private seed over $\mathbb{Z}_p$, denoted as, sPRG $\in (\mathbb{C}, \deg d)$, if for every I in the support of $\mathsf{IdSamp}(1^\lambda, 1^n)$, there exists an algorithm $\mathsf{Process}_I$ in $\mathbb{C}$ and an $m(n)$-tuple of polynomials Q_I that can be efficiently generated from I, such that for all sd in the support of $\mathsf{SdSamp}(I)$, it holds that:

$$\mathsf{Eval}(I, \mathsf{sd}) = Q_I(\overline{P}, S) \text{ over } \mathbb{Z}_p \ , \ \overline{P} = \mathsf{Process}_I(P) \ ,$$

where Q_I has degree 1 in $\overline{P}$ and degree d in S.

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

Definition 3.4 (Pseudo-Random 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 $S \leftarrow \{0,1\}^n$, where the polynomial poly is fixed to be 1.

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 $\mathsf{Eval}(I, \mathsf{sd})$ 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 $\mathsf{Eval}(I, \mathsf{sd})$ depends on at most c bits of S.

4 CONSTRUCTION OF STRUCTURED SEED PRG

In this section, we construct a family of structured-seed PRGs whose evaluation has degree 2 in the private seed, and constant degree in the public seed.

Theorem 4.1. *Let λ be the security parameter. Let $d \in \mathbb{N}, \delta \in (0,1), \tau > 1$ be arbitrary constants.*

Assuming the following:

- *the existence of a constant locality Boolean* PRG *with stretch $\tau > 1$ and multilinear degree d over $\mathbb{Z}$, and,*
- *δ-$\mathsf{LPN}_{\mathbb{F}}$-assumption holds,*

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

Technical Overview. Let PRG = (IdSamp, Eval) be the Boolean PRG with multilinear degree d and stretch τ. Our sPRG will simply evaluate PRG on an input $\boldsymbol{\sigma} \in \{0,1\}^n$ and return its output $\boldsymbol{y} \in \{0,1\}^m$ where $m = n^\tau$. The challenge stems from the fact that the evaluation algorithm $\mathsf{Eval}_I(\boldsymbol{\sigma})$ of PRG has degree d in its private seed $\boldsymbol{\sigma}$, but the evaluation algorithm $\mathsf{Eval}'_I(P, S)$ of sPRG can only have degree 2 in the private seed S. To resolve this, we pre-process $\boldsymbol{\sigma}$ into appropriate public and private seeds (P, S) and leverage the LPN over $\mathbb{Z}_p$ assumption to show that the seed is hidden.

Towards this, sPRG uses a λ-bit prime p and "encrypts" the seed $\boldsymbol{\sigma}$ using LPN samples over $\mathbb{Z}_p$ as follows:

$$\text{Sample: } \mathbf{A} \leftarrow \mathbb{Z}_p^{\ell \times n}, \ \boldsymbol{s} \leftarrow \mathbb{Z}_p^{1 \times \ell}, \ \boldsymbol{e} \leftarrow \mathcal{D}_r^{1 \times n}(p)$$

Add to the function index I': $\mathbf{A}$

Add to public seed P: $\boldsymbol{b} = \boldsymbol{s}\mathbf{A} + \boldsymbol{e} + \boldsymbol{\sigma}$

It follows directly from the LPN over $\mathbb{Z}_p$ assumption that $(\mathbf{A}, \boldsymbol{b})$ 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 a $\ell^{-\delta}$ fraction of components – thus it is a sparsely erroneous version of the seed.

Given such "encryption", by applying previous techniques [6, 68, 89, 90] that work essentially by "replacing monomials" (previous works replace monomials in the PRG seed with polynomials in the LWE secret, and we here replace the monomials in the erroneous seed with polynomials in the LPN secret) we can compute PRG on the erroneous seed $\boldsymbol{\sigma} + \boldsymbol{e}$ via a polynomial map $G^{(1)} = (G_1^{(1)}, \ldots, G_m^{(1)})$ that depends on $\mathbf{A}$ and has degree d in the public component $\boldsymbol{b}$ and only degree 2 in all possible degree $\lceil \frac{d}{2} \rceil$ monomials in $\boldsymbol{s}$. More precisely, each polynomial $G_j^{(1)}$ denotes the polynomial computing the j^{th} output coordinate and is defined as follows:

$$G^{(1)}\left(\boldsymbol{b}\ ,\ (\bar{\boldsymbol{s}}^{\otimes \lceil \frac{d}{2} \rceil})\right) \coloneqq \mathsf{Eval}_I(\boldsymbol{b} - \boldsymbol{s}\mathbf{A}) = \mathsf{Eval}_I\left(\boldsymbol{\sigma} + \boldsymbol{e}\right),\ \bar{\boldsymbol{s}} = \boldsymbol{s} || 1 \quad (1)$$

where for any vector $\boldsymbol{v}$, we denote by $\boldsymbol{v}^{\otimes k}$ the tensoring of the vector $\boldsymbol{v}$ with itself k times, yielding a vector of dimension $\dim(\boldsymbol{v})^k$. In particular, observe that by setting the dimension ℓ of secret $\boldsymbol{s}$ sufficiently small, the polynomial map $G^{(1)}$ can be expanding; namely, set $\ell(n)$ so that $n' = \dim((\boldsymbol{b}, \bar{\boldsymbol{s}}^{\otimes \lceil \frac{d}{2} \rceil})) = \left(n + \ell^{\lceil \frac{d}{2} \rceil}\right) \ll m = n^\tau$. In this overview, we compare the number of output bits m with the number of field elements in the seed of sPRG because we will choose a λ-bit modulus p. Thus, the seed length is $n'\lambda$, with $n' \ll m$, satisfying the notion of polynomial stretch for sPRG (in particular, we have polynomial expansion as λ is asymptotically smaller than n' and m).

However, a new problem arises: even though the degree fits, $G^{(1)}$ only evaluates an erroneous output $\boldsymbol{y}' = \mathsf{Eval}_I(\boldsymbol{\sigma} + \boldsymbol{e})$, but we want to obtain the correct output $\boldsymbol{y} = \mathsf{Eval}_I(\boldsymbol{\sigma})$. To correct errors, we further modify the polynomial and include more pre-processed information in the private seeds. Our key observation is the following: Because LPN noises are sparse, and because Eval_I has only constant locality, only a few outputs depend on erroneous seed locations. We refer to them as bad outputs and let BAD denote the set of their indices. The probability that a single output bit is bad is bounded by $\frac{O(1)}{\ell^\delta}$. By a simple Markov argument, the number of bad outputs is bounded by $\Gamma = \frac{m \log n}{\ell^\delta}$ with probability $1 - o(1)$. Leveraging this sparsity, we devise a method for correcting the bad outputs below.

We describe a sequence of ideas that lead to the final correction method. To start with, we describe two wrong ideas that illustrate the challenges we will overcome. Then, we show how to correct bad outputs in three simple cases that are increasingly less restrictive, which lead to our final solution.

Challenges to Correction:

- The first wrong idea is correcting by adding the difference $\mathrm{Corr} = \boldsymbol{y} - \boldsymbol{y}'$ between the correct and erroneous outputs, $\boldsymbol{y} = \mathrm{Eval}_I(\boldsymbol{\sigma})$ and $\boldsymbol{y}' = \mathrm{Eval}_I(\boldsymbol{\sigma} + \boldsymbol{e})$; we refer to Corr as the *correction vector*. To obtain the correct output, evaluation can compute the polynomial map $G^{(1)}\left(\boldsymbol{b}\,,\ (\overline{\boldsymbol{s}}^{\otimes \lceil \frac{d}{2} \rceil})\right) + \mathrm{Corr}$. The problem is that Corr must be included in the seed, but it is as long as the output and would kill expansion.
- To fix expansion, the second wrong idea is adding correction only for bad outputs, so that the seed only stores non-zero entries in Corr, which is short (bounded by Γ elements). More precisely, the j'th output can be computed as $G_j^{(1)}\left(\boldsymbol{b}\,,\ (\overline{\boldsymbol{s}}^{\otimes \lceil \frac{d}{2} \rceil})\right) + \mathrm{Corr}_j$ 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 2 polynomial in the private seed.

These two wrong ideas illustrate the first difficulty: 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 2 computation on the private seed. The second challenge stems from the fact that with noticeable probability, the number of bad outputs is not bounded by Γ. We will overcome this via parallel repetition: Instead of having one LPN encryption of the seed, we will have λ copies $\boldsymbol{b}^{(k)} = \boldsymbol{s}^{(k)}\mathbf{A} + \boldsymbol{e}^{(k)} + \boldsymbol{\sigma}$, so that with overwhelming probability at least one copy has sparse bad outputs. This copy can be used to compute the output of the PRG. However, we must be careful to hide which copies have sparse bad outputs.

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[4], 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 $\mathrm{Corr} = \mathbf{UV}$, and compute the correct output as follows:

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

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.

Simple Case 2: Up to Γ bad outputs, which are however evenly spread. The above method however cannot handle more than $\sqrt{m}$ bad outputs, whereas the actual number of bad outputs can be up to $\Gamma = m(\log n)/\ell^\delta$, 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 $\log n$ bad outputs in each block. 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 $\ell^{\delta/2} \times \log n$ and $\log n \times \ell^{\delta/2}$ respectively, and correct bad outputs as follows:

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

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 $\ell^{\delta/2} \log n$ field elements, and the total number of elements is $\ell^{\delta/2} \log n \cdot \frac{m}{\ell^\delta}$ 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.

At this point, it is tempting to wish that bad outputs must be evenly spread given that the LPN noises occur at random locations. This is, however, not true because the input-output dependency graph of PRG is arbitrary, and the location of bad outputs are correlated. Consider the example that the PRG output dependency graph is such that for the first $K = \ell^\delta \cdot \log^2 n$ seed bits, the i'th seed bit impacts all the output positions in the i'th block. In this case, with overwhelming probability, one of the first K seed bits would be errorneous, and hence at least one block will be completely corrupt.

Simple Case 3: Up to Γ bad outputs. To overcome this, our next idea is to "force" the even spreading of the bad outputs, by assigning output bits randomly into B buckets, and then compress the correction vector corresponding to each bucket.

Step 1: Randomly assign outputs. We assign the outputs into B buckets, via a random mapping $\phi_{\mathsf{bkt}} : [m] \to [B]$. The number of buckets is set to $B = mt/\ell^\delta$ where t is a *slack parameter* set to λ. By a Chernoff-style argument, we can show that each bucket contains at most $t^2\ell^\delta$ output bits, and at most t of them are bad, except with negligible probability in $t = \lambda$. As such, bad outputs are evenly spread among a small number of not-so-large buckets.

Step 2: Compress the buckets. Next, we organize each bucket i into a matrix $\mathbf{M}_i$ of dimension $t\ell^{\delta/2} \times t\ell^{\delta/2}$ and then compute its factorization $\mathbf{M}_i = \mathbf{U}_i\mathbf{V}_i$ with respect to matrices of dimensions $t\ell^{\delta/2} \times t$ and $t \times t\ell^{\delta/2}$ respectively. To form matrix $\mathbf{M}_i$, we use another mapping $\phi_{\mathsf{ind}} : [m] \to [t\ell^{\delta/2}] \times [t\ell^{\delta/2}]$ 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; other than that, it can be arbitrary. $(\mathbf{M}_i)_{k,l}$ is set to Corr_j if there is j such that $\phi_{\mathsf{bkt}}(j) = i$ and $\phi_{\mathsf{ind}}(j) = (k, l)$, and set to 0 if no such j exists. Since every matrix $\mathbf{M}_i$ has at most t non-zero entries, we can factor them and compute

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

the correct output as:

$$\forall j \in [m], G_j^{(2)}\Big(\boldsymbol{b}\,,\ \underbrace{\Big(\overline{\boldsymbol{s}}^{\otimes\lceil\frac{d}{2}\rceil},\ (\mathbf{U}_i,\mathbf{V}_i)_{i\in[B]}\Big)}_{S}\Big)$$
$$= G_j^{(1)}\left(\boldsymbol{b}\,,\ (\overline{\boldsymbol{s}}^{\otimes\lceil\frac{d}{2}\rceil})\right) + (\mathbf{U}_{\phi_{\mathsf{bkt}}(j)}\cdot\mathbf{V}_{\phi_{\mathsf{bkt}}(j)})_{\phi_{\mathsf{ind}}(j)}\ ,$$

$G^{(2)}$ is expanding, because the number of field elements in $\mathbf{U}_i$'s and $\mathbf{V}_i$'s are much smaller than m, namely: $2t^2\ell^{\delta/2}B = O(m\lambda^3/\ell^{\delta/2}) \ll m$ (recall that $t=\lambda$), if ℓ is a large enough polynomial in λ. Note that it is important that the assignments ϕ_{bkt} and ϕ_{ind} are not included in the seed as their description is as long as the output. Fortunately, they are reusable and can be included in the function index $I' = (I, \mathbf{A}, \phi_{\mathsf{bkt}}, \phi_{\mathsf{ind}})$.

Final Case: More than Γ bad outputs with noticable probability. Unfortunately, the number of bad output bits are only bounded by Γ with probability noticably away from 1. This is inevitable: Consider the example that every output bit of PRG depends on the first seed bit, and with inverse polynomial probability $\frac{1}{\ell^\delta}$, it is erroneous and so are all outputs. When bad outputs are not sparse, our analysis above does not apply, in particular, some bucket may be assigned more than $t=\lambda$ bad outputs. We resort to parallel repetition to deal with this. Instead of generating a single LPN encryption of the seed $\boldsymbol{\sigma}$, generate λ instances, $\left\{\boldsymbol{b}^{(k)} = \boldsymbol{s}^{(k)}\mathbf{A} + \boldsymbol{\sigma} + \boldsymbol{e}^{(k)}\right\}_{k\in[\lambda]}$. For each instance k, we can compute the corresponding correction vector $\mathsf{Corr}^{(k)}$ and set a flag $\mathsf{flag}^{(k)} = 1$ if k is the *smallest* index such that the number of bad outputs is bounded by Γ ($\mathsf{flag}^{(k)} = 0$ otherwise). By that fact that each instance satisfies the desired sparsity with high probability and is independent, with all but exponentially small probability, some flag is set $\mathsf{flag}^{(k^\star)} = 1$ and $k^\star$ is unique. If no flag is set, we set $\mathsf{err} = 1$, which occurs with exponentially small probability ($\mathsf{err} = 0$ otherwise). For the special $k^\star$, we can compress its correction vector $\mathsf{Corr}^{k^\star}$ into $(\mathbf{U}_i^{(k^\star)}, \mathbf{V}_i^{(k^\star)})_{i\in[B]}$ and set the secret seed $S^{(k^\star)}$ as before, which satisfies:

$$G^{(2)}\Bigg(\boldsymbol{b}^{(k^\star)}\,,\ \underbrace{\left(\left(\overline{\boldsymbol{s}}^{(k^\star)}\right)^{\otimes\lceil\frac{d}{2}\rceil},\ \left(\mathbf{U}_i^{(k^\star)}, \mathbf{V}_i^{(k^\star)}\right)_{i\in[B]}\right)}_{S^{(k^\star)}}\Bigg) = \mathsf{Eval}_I(\boldsymbol{\sigma})\ .$$

The tricky part is that we must hide the value of $k^\star$ as it leaks a single bit of information about the noise vector $\boldsymbol{e}^{(k^\star)}$, namely, it leads to sparse bad outputs. To do so, we set $S^{(k^\star)} = \mathbf{0}$ for every other instance $k \neq k^\star$, and aim to compute the boxed part the in following equation.

$$G^{(3)}\Bigg(\underbrace{\left(\left\{\boldsymbol{b}^{(k)}\right\}_k, \mathsf{err}\right)}_{P'}, \underbrace{\left\{\mathsf{flag}^{(k)}, S^{(k)}, \mathsf{flag}^{(k)}\cdot S^{(k)}\right\}_k}_{S'}\Bigg)$$
$$:= \boxed{(1-\mathsf{err})\left(\textstyle\sum_{k\in[\lambda]}\mathsf{flag}^{(k)}\cdot G^{(2)}\left(\boldsymbol{b}^{(k^\star)}\,,\ S^{(k^\star)}\right)\right)}$$
$$= (1-\mathsf{err})\mathsf{Eval}_I(\boldsymbol{\sigma})\ .$$

The last equality holds since when $\mathsf{err}=0$, there is a unique $k^\star$ for which the flag is 1. The boxed part can be computed by a polynomial $G^{(3)}$ with degree $d+1$ in the public seed P' and degree 2 in the private seed S', since the multiplication between $\mathsf{flag}^{(k)}$ and $S^{(k)}$ is pre-computed in S'. Furthermore, we observe that comparing with polynomial $G^{(2)}$, the length of the seed for $G^{(3)}$ has increased by a factor of $O(\lambda)$, which does not hurt expansion.

For security, observe that the polynomial map $G^{(3)}$ is oblivious of LPN noises in all instances, while the public seed leaks the error flag err, which is 1 with only exponentially small probability. Therefore, by the LPN over $\mathbb{Z}_p$ assumption, the seed $\boldsymbol{\sigma}$ of PRG is hidden in all λ instances and the security of PRG ensures that the output is pseudorandom. We now proceed to the formal construction and proof.

Construction. Assume the premise of the theorem. Let (IdSamp, Eval) be the function index sampling algorithm and evaluation algorithm for the 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 security parameter λ,
- the seed length parameter n' of the sPRG and output length $m' = (n')^\tau$,
- the stretch Γ and degree d of PRG, where Γ can be an arbitrary positive constant greater than 1, and d can be an arbitrary positive integer,
- the input length parameter n and output length $m = n^\tau$ of the PRG (as we see below, by construction, $m' = m$ and n' is larger than n, but still sublinear in m'),
- the LPN secret dimension $\ell = n^{1/\lceil d/2\rceil}$ and modulus p, which is a λ bit prime,
- a threshold $\Gamma = m\cdot\log n/\ell^\delta$ of the number of bad outputs that can be tolerated,
- a slack parameter t used for bounding the capacity of each bucket and number of bad outputs in each bucket, which is set to $t=\lambda$,
- a parameter $B = m\cdot t/\ell^\delta$ that indicates the number of buckets used, and
- a parameter $c = t^2\ell^\delta$ that indicates the capacity of each bucket.

$I' \leftarrow \mathsf{IdSamp}'(1^\lambda, 1^{n'})$**:** (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 4.1 for the exact relationship between n and n'.)
Sample $I \leftarrow \mathsf{IdSamp}(1^\lambda, 1^n)$ and $\mathbf{A} \leftarrow \mathbb{Z}_p^{\ell\times n}$. Prepare two functions $\boldsymbol{\phi} = (\phi_{\mathsf{bkt}}, \phi_{\mathsf{ind}})$ as follows:

- Sample a random function $\phi_{\mathsf{bkt}} : [m] \to [B]$ mapping every output location to one of B buckets. Let $\phi_{\mathsf{bkt}}^{-1}(i)$ for $i\in[B]$ denote the set of preimages of i through ϕ_{bkt}. This set contains all outputs assigned to the bucket i.
- Prepare $\phi_{\mathsf{ind}} : [m] \to [\sqrt{c}]\times[\sqrt{c}]$ in two cases:
  - *If some bucket exceeds capacity*, that is, there exists $i\in[B]$ such that $|\phi_{\mathsf{bkt}}^{-1}(i)| > c$, set ϕ_{ind} to be a constant function always outputting $(1,1)$.
  - *Otherwise if all buckets are under capacity*, for every index $j\in[m]$, ϕ_{ind} maps j to a pair of indexes $(k_j, l_j) \in$

$[\sqrt{c}] \times [\sqrt{c}]$, under the constraint that two distinct output bits $j_1 \neq j_2$ that are mapped into the same bucket $\phi_{\mathsf{bkt}}(j_1) = \phi_{\mathsf{bkt}}(j_2)$ must have distinct pairs of indices $\phi_{\mathsf{ind}}(j_1) \neq \phi_{\mathsf{ind}}(j_2)$.

Output $I' = (I, \boldsymbol{\phi} = (\phi_{\mathsf{bkt}}, \phi_{\mathsf{ind}}), \mathbf{A})$.

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

- Sample a PRG seed $\boldsymbol{\sigma} \leftarrow \{0,1\}^n$.
- For every $k \in [\lambda]$, run $(\mathsf{flag}^{(k)}, P^{(k)}, S^{(k)}) \leftarrow \mathsf{1SdSamp}(I', \boldsymbol{\sigma})$, where the 1SdSamp algorithm is described below.
- Set $\mathsf{err} = 1 - \max_{k\in[\lambda]}(\mathsf{flag}^{(k)})$. That is err is 1 if no flag is set, and otherwise, it is zero.
- If $\mathsf{err} = 0$, let $k^\star = \min\{k \ : \ \mathsf{flag}^{(k)} = 1\}$ and for every $k > k^\star$, overwrite $\mathsf{flag}^k = 0$. (This ensure that there is a unique k for which the flag is set to 1.)
- Set $P' = (\{P^{(k)}\}_{k\in[\lambda]}, \mathsf{err})$ and $S' = (\{\mathsf{flag}^{(k)}, S^{(k)}, \mathsf{flag}^{(k)} \cdot S^{(k)}\}_{k\in[\lambda]})$.

Output $\mathsf{sd} = (P', S')$,

Sampling Algorithm $\mathsf{1SdSamp}(I', \boldsymbol{\sigma})$

Input: sPRG index $I' = (I, \boldsymbol{\phi} = (\phi_{\mathsf{bkt}}, \phi_{\mathsf{ind}}), \mathbf{A})$ and random seed $\boldsymbol{\sigma}$.

Output: A flag $\mathsf{flag} \in \{0,1\}$, and a pair of public and private seeds (P, S).

- 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
$$\boldsymbol{b} = \boldsymbol{s}\mathbf{A} + \boldsymbol{\sigma} + \boldsymbol{e} \ .$$
- 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 j'th output of Eval_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 flag = 0 if
  (1) *Too many* bad *output bits:* $|\mathsf{BAD}| > \Gamma$,
  (2) **or** *Some bucket exceeds capacity:* $\exists i \in [B], |\phi_{\mathsf{bkt}}^{-1}(i)| > c$,
  (3) **or** *Some bucket contains too many* bad *outputs:* $\exists i \in [B], |\phi_{\mathsf{bkt}}^{-1}(i) \cap \mathsf{BAD}| > t$.
  Otherwise, set flag = 1.
- Compute the outputs of PRG on input the correct seed and the erroneous seed, $\boldsymbol{y} = \mathsf{Eval}_I(\boldsymbol{\sigma})$ and $\boldsymbol{y}' = \mathsf{Eval}_{\boldsymbol{I}}(\boldsymbol{\sigma}+\boldsymbol{e})$. Set the correction vector $\mathsf{Corr} = \boldsymbol{y} - \boldsymbol{y}'$.
- Construct matrices $\mathbf{M}_1, \ldots, \mathbf{M}_B$, by setting
$$\forall j \in [m], \ \left(\mathbf{M}_{\phi_{\mathsf{bkt}}(j)}\right)_{\phi_{\mathsf{ind}}(j)} = \mathsf{Corr}_j$$
Every other entry is set to 0.
- "Compress" matrices $\mathbf{M}_1, \ldots, \mathbf{M}_B$ as follows:
  – If flag = 1, for every $i \in [B]$ compute factorization
  $$\mathbf{M}_i = \mathbf{U}_i\mathbf{V}_i, \qquad \mathbf{U}_i \in \mathbb{Z}_p^{\sqrt{c}\times t}, \ \mathbf{V}_i \in \mathbb{Z}_p^{t\times\sqrt{c}}$$
  This factorization exists because when flag = 1, condition 3 above implies that each $\mathbf{M}_i$ has at most t nonzero entries, and hence rank at most t.
  – If flag = 0, for every $i \in [B]$, set $\mathbf{U}_i$ and $\mathbf{V}_i$ to be 0 matrices.
- Set the public seed to $P = \boldsymbol{b}$.
- Prepare the private seed S as follows. Let $\bar{\boldsymbol{s}} = \boldsymbol{s}||1$.
$$S = \left(\bar{\boldsymbol{s}}^{\otimes\lceil\frac{d}{2}\rceil}, \{\mathbf{U}_i, \mathbf{V}_i\}_{i\in[B]}\right) \tag{2}$$

Output (flag, P, S) as $\mathbb{Z}_p$ elements.

$\boldsymbol{y}' \leftarrow \mathsf{Eval}'(I', \mathsf{sd})$**:** Compute $\boldsymbol{y} \leftarrow \mathsf{Eval}(I, \boldsymbol{\sigma})$, and output $\boldsymbol{z} = (1 - \mathsf{err}) \cdot \boldsymbol{y}$. This computation is done via a polynomial map $G^{(4)}$, such that,

$$G^{(4)}(P', S') = \sum_{k\in[\lambda]} G^{(3)}\left(P^{(k)}, \left(\mathsf{flag}^{(k)}, S^{(k)}, \mathsf{flag}^{(k)} \cdot S^{(k)}\right)\right) = \sum_{k\in[\lambda]} \mathsf{flag}^{(k)} \cdot \mathsf{Eval}_I(\boldsymbol{\sigma}) = (1 - \mathsf{err}) \cdot \boldsymbol{y} \ .$$

where the second equality follows from the fact that when err = 0, there is a unique $k \in [\lambda]$ such that $\mathsf{flag}^{(k)} = 1$, and when err = 1, all flags are zero. Furthermore, the polynomial mapping $G^{(3)}$ computing $\mathsf{flag}^{(k)} \cdot \mathsf{Eval}_I(\boldsymbol{\sigma})$ is described below. $G^{(3)}$ has constant degree d in the public seed $P^{(k)}$ and only degree 2 in the private seed $(\mathsf{flag}^{(k)}, S^{(k)}, \mathsf{flag}^{(k)} \cdot S^{(k)})$. Hence $G^{(4)}$ also has constant degree d in P' and degree 2 in S'.

The Polynomial $G^{(3)}$ $(P, (\mathsf{flag}, S, \mathsf{flag} \cdot S))$

Constant: $I' = (I, \boldsymbol{\phi} = (\phi_{\mathsf{bkt}}, \phi_{\mathsf{ind}}), \mathbf{A})$

Input: $P = \boldsymbol{b} = \boldsymbol{s}\mathbf{A} + \boldsymbol{\sigma} + \boldsymbol{e}$, $S = (\bar{\boldsymbol{s}}^{\otimes\lceil\frac{d}{2}\rceil}, \{\mathbf{U}_i, \mathbf{V}_i\}_{i\in[B]})$

Output: $\mathsf{flag} \cdot \mathsf{Eval}_I(\boldsymbol{\sigma})$.

We describe $G^{(3)}$ using intermediate polynomials $G^{(1)}$ and $G^{(2)}$.

- 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 $\boldsymbol{\sigma}$).
  Notation Let us introduce some notation for monomials. A monomial h on a vector $\boldsymbol{a}$ is represented by the set of indices $h = \{i_1, i_2, \ldots, i_k\}$ of variables used in it. h evaluated on $\boldsymbol{a}$ 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
  $$\forall j \in [m], \ \ y_j = \mathsf{Eval}_j(\boldsymbol{\sigma}) = L_j((\sigma_h)_{h\in\mathsf{Eval}_j}),$$
  for a linear L_j.

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

$$x_i = \langle \boldsymbol{c}_i, \bar{\boldsymbol{s}} \rangle,$$
$$x_v = \langle \otimes_{i \in v} \boldsymbol{c}_i, \otimes_{i \in v} \bar{\boldsymbol{s}} \rangle,$$

where, $\boldsymbol{c}_i = -\boldsymbol{a}_i^{\mathrm{T}} || b_i$ and $\boldsymbol{a}_i$ is the ith column of $\mathbf{A}$. (Recall that $\otimes_{i \in v} \boldsymbol{z}_i = \boldsymbol{z}_{i_1} \otimes \cdots \otimes \boldsymbol{z}_{i_k}$ if $v = \{i_1, \ldots, i_k\}$ and is not empty; otherwise, it equals 1.) Combining with the previous step, we obtain a polynomial mapping $G^{(1)}(\boldsymbol{b}, S)$ that computes $\mathsf{Eval}(\boldsymbol{\sigma} + \boldsymbol{e})$:

$$G_j^{(1)}(\boldsymbol{b}, S) \coloneqq L_j \left((\langle \otimes_{i \in v} \boldsymbol{c}_i, \otimes_{i \in v} \bar{\boldsymbol{s}} \rangle)_{v \in \mathsf{Eval}_j} \right) . \quad (3)$$

Note that $G^{(1)}$ implicitly depends on $\mathbf{A}$ 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 $\bar{\boldsymbol{s}}^{\otimes \lceil \frac{d}{2} \rceil}$ (see Equation (1)), and hence $S \otimes S$ contains all monomials in $\boldsymbol{s}$ 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 $\boldsymbol{y} = \mathsf{Eval}(\boldsymbol{\sigma})$ on all good output bits.

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

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

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

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 three conditions for $\mathsf{flag} = 1$, every entry j in the correction vector Corr_j is placed at entry $\left(\mathbf{M}_{\phi_{\mathsf{bkt}}(j)} \right)_{\phi_{\mathsf{ind}}(j)}$. 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 can be done by simply multiplying flag to the output of $G^{(2)}$, giving the final polynomial

$$G^{(3)}(P, S) = \mathsf{flag} \cdot G^{(2)}(P, S) . \quad (4)$$

At last, $G^{(3)}$ still has degree d in the public seed, and only degree 2 in the private seed, since multiplication between flag and S is precomputed.

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 a λ-bit prime, and the slack parameter $t = \lambda$.

Claim 4.1. *For the parameters as set in the Construction,* sPRG *has stretch of τ' for some constant $\tau' > 1$.*

Proof. Let's start by analyzing the length of the public and private seeds.

- The public seed contains $P' = (\{\boldsymbol{b}^{(k)}\}_{k \in [\lambda]}, \mathsf{err})$, where

$$\left| \boldsymbol{b}^{(k)} \right| = O(n \log p) = O(n \cdot \lambda) .$$

- The private seed S' contains $\{\mathsf{flag}^{(k)}, S^{(k)}, \mathsf{flag}^{(k)} \cdot S^{(k)}\}$, where each $S^{(k)}$ contains the following

$$S_1^{(k)} = \left(\bar{\boldsymbol{s}}^{(k)} \right)^{\otimes \lceil \frac{d}{2} \rceil}, \qquad S_2^{(k)} = \left\{ \mathbf{U}_i^{(k)}, \mathbf{V}_i^{(k)} \right\}_{i \in [B]} .$$

The bit-lengths are:

$$\begin{aligned}
\left| S_1^{(k)} \right| &= (\ell + 1)^{\lceil d/2 \rceil} \log p \\
&= O\left(n^{\frac{1}{\lceil d/2 \rceil}} \right)^{\lceil d/2 \rceil} \log p = O(n \cdot \lambda) \\
&\quad (\text{ by } \ell = n^{\lceil d/2 \rceil},\ \log p = \lambda) \\
\left| S_2^{(k)} \right| &= 2B \cdot t \cdot \sqrt{c} \cdot \log p \\
&= \frac{2mt}{\ell^\delta} \cdot t \cdot t\ell^{\delta/2} \cdot \log p = \frac{2mt^3 \log p}{\ell^{\delta/2}} \\
&\quad (\text{by } B = \frac{mt}{\ell^\delta},\ c = t^2 \ell^\delta) \\
&= \frac{2m\lambda^4}{\ell^{\delta/2}} \\
&\quad (\text{ by } t = \log p = \lambda)
\end{aligned}$$

Because $\ell^{\delta/2} = n^{\frac{\delta}{2\lceil \frac{d}{2} \rceil}}$, $m = n^\tau$, and $k \in [\lambda]$, we have:

$$|\mathsf{sd}| = 1 + \sum_{k \in [\lambda]} \left(\left| \boldsymbol{b}^{(k)} \right| + 2 \left| S_1^{(k)} \right| + 2 \left| S_2^{(k)} \right| + 1 \right)$$
$$= O((n + n^{\tau - \frac{\delta}{2\lceil \frac{d}{2} \rceil}}) \cdot \lambda^5)$$

We set $n' = O(n + n^{\tau - \frac{\delta}{2\lceil \frac{d}{2} \rceil}})$, therefore $m = n'^{\tau'}$ for some $\tau' > 1$ and the seed length is $n' \mathsf{poly}(\lambda)$. This concludes the proof. □

Proof of Pseudorandomness. We prove the following proposition which implies that sPRG is pseudorandom and the security loss from the Boolean PRG and the $\mathsf{LPN}_{\mathbb{F}}$ assumption is polynomial in λ. Therefore, our sPRG is polynomially or subexponentially secure, if the underlying Boolean PRG and the $\mathsf{LPN}_{\mathbb{F}}$ assumptions are polynomially or subexponentially secure respectively.

Proposition 4.2. *For parameters as set in the Construction, if δ-$\mathsf{LPN}_{\mathbb{F}}$ is ϵ-hard, and* PRG *is ϵ-pseudorandom, the following two distributions are γ-indistinguishable, where $\gamma = O(\lambda\epsilon)$.*

$$\mathcal{H}_0 = \left\{ (I, \boldsymbol{\phi}, \mathbf{A}, \{\boldsymbol{b}^{(k)}\}_{k \in [\lambda]}, \mathsf{err}, \boldsymbol{z}) \right\}$$
$$\mathcal{H}_3 = \left\{ (I, \boldsymbol{\phi}, \mathbf{A}, \{\boldsymbol{b}^{(k)}\}_{k \in [\lambda]}, \mathsf{err}, \boldsymbol{r}) \right\},$$

where $I' \leftarrow \mathsf{IdSamp}'(1^\lambda, 1^{n'})$, $(P', S') \leftarrow \mathsf{SdSamp}'(I')$, $\boldsymbol{z} \leftarrow \mathsf{Eval}'(I', \mathsf{sd})$, and $\boldsymbol{r} \leftarrow \{0,1\}^m$. (Recall that $I' = (I, \boldsymbol{\phi}, \mathbf{A})$, $P' = (\{\boldsymbol{b}^{(k)}\}_{k \in [\lambda]}, \mathsf{err})$.)

Proof. We show that $\mathcal{H}_0$ and $\mathcal{H}_3$ are γ-indistinguishable via the following two hybrids.

$$\mathcal{H}_1 = \left\{ (I, \boldsymbol{\phi}, \mathbf{A}, \{\boldsymbol{b}^{(k)}\}_{k\in[\lambda]}, \boxed{0, \boldsymbol{y}}) \right\}$$
$$\mathcal{H}_2 = \left\{ (I, \boldsymbol{\phi}, \mathbf{A}, \{\boldsymbol{b}^{(k)}\}_{k\in[\lambda]}, \boxed{0}, \boldsymbol{r}) \right\},$$

where $I' \leftarrow \mathsf{IdSamp}'(1^\lambda, 1^{n'})$, $(P', S') \leftarrow \mathsf{SdSamp}'(I')$, $\boldsymbol{y} \leftarrow \mathsf{Eval}(I, \boldsymbol{\sigma})$, and $\boldsymbol{r} \leftarrow \{0,1\}^m$. Observe that the only difference between $\mathcal{H}_0$ and $\mathcal{H}_1$ (highlighted in the box) is that in the former err is the flag set by the SdSamp' algorithm and the sPRG output $\boldsymbol{z} = (1 - \mathsf{err})\boldsymbol{y}$, whereas in the latter err is replaced with a constant 0 and the output is replaced with $\boldsymbol{y}$. Similarly, the only difference between $\mathcal{H}_2$ and $\mathcal{H}_3$ (highlighted in the box) is whether err is replaced with 0. We show in Claim 4.2 that the probability that err = 1 is exponentially small.

Claim 4.2. *For parameters as set in the Construction, it holds that* $\Pr[\ \mathsf{err} = 1\ ] \leq 2^{-\lambda}$, *where* $\mathsf{IdSamp}'(1^\lambda, 1^{n'})$, *and* $(P' = (\{\boldsymbol{b}^{(k)}\}_{k\in[\lambda]}, \mathsf{err}), S') \leftarrow \mathsf{SdSamp}'(I')$.

Therefore, the statistical distance between $\mathcal{H}_0$ and $\mathcal{H}_1$, as well as $\mathcal{H}_2$ and $\mathcal{H}_3$, is bounded by $2^{-\lambda}$. Due to space limit, we defer the proof to the full version [91].

It remains to show that $\mathcal{H}_1$ and $\mathcal{H}_2$ are $O(\lambda\epsilon)$-indistinguishable. This follows from the facts that *i)* $\boldsymbol{\phi}$ is completely independent of $(I, \mathbf{A}, \{\boldsymbol{b}^{(k)}\}_{k\in[\lambda]}, \boldsymbol{y})$ or $(I, \mathbf{A}, \{\boldsymbol{b}^{(k)}\}_{k\in[\lambda]}, \boldsymbol{r})$, and *ii)* $(I, \mathbf{A}, \{\boldsymbol{b}^{(k)}\}_{k\in[\lambda]}, \boldsymbol{y})$ and $(I, \mathbf{A}, \{\boldsymbol{b}^{(k)}\}_{k\in[\lambda]}, \boldsymbol{r})$ are indistinguishable following from the LPN assumption and the pseudorandomness of PRG. This indistinguishability is is the heart of the security of sPRG, and is captured in Lemma 4.3 below.

Lemma 4.3. *For parameters set as in the construction, assuming that* $(\mathsf{IdSamp}, \mathsf{Eval})$ *is a* ϵ*-secure Boolean pseudorandom generator and* δ-$\mathsf{LPN}_{\mathbb{F}}$ *is* ϵ*-hard, the following two distributions are* $O(\lambda\epsilon)$*-indistinguishable:*

$$\mathcal{D}_0 = \left\{ I,\ \boldsymbol{A},\ \{\boldsymbol{b}^{(k)} = \boldsymbol{s}^{(k)} \cdot \boldsymbol{A} + \boldsymbol{e}^{(k)} + \boldsymbol{\sigma}\}_{k\in[\lambda]},\ \boldsymbol{y} \right\}$$
$$\mathcal{D}_{\lambda+1} = \left\{ I,\ \boldsymbol{A},\ \{\boldsymbol{u}^{(k)}\}_{k\in[\lambda]},\ \boldsymbol{w} \right\}$$

where $\boldsymbol{A} \leftarrow \mathbb{Z}_p^{\ell\times n}$, $\boldsymbol{s}^{(k)} \leftarrow \mathbb{Z}_p^{1\times\ell}$, $\boldsymbol{e}^{(k)} \leftarrow \mathcal{D}_r^{1\times n}(p)$, $\boldsymbol{\sigma} \leftarrow \{0,1\}^{1\times n}$, $I \leftarrow \mathsf{IdSamp}(1^\lambda, 1^n)$, $\boldsymbol{y} \leftarrow \mathsf{Eval}(I, \boldsymbol{\sigma})$, $\boldsymbol{u} \leftarrow \mathbb{Z}_p^{1\times n}$, *and* $\boldsymbol{w} \leftarrow \{0,1\}^{1\times m}$.

Proof. We introduce intermediate distributions $\mathcal{D}_i$ defined as follows:

$$\mathcal{D}_i = \left\{ I,\ \boldsymbol{A},\ \{\boldsymbol{u}^{(k)}\}_{k\in[i]},\ \{\boldsymbol{b}^{(k)} = \boldsymbol{s} \cdot \boldsymbol{A} + \boldsymbol{e}^{(k)} + \boldsymbol{\sigma}\}_{i<k\leq\lambda},\ \boldsymbol{y} \right\}$$

where the first i LPN samples $\boldsymbol{b}^{(1)} \cdots \boldsymbol{b}^{(i)}$ are replaced with random strings $\boldsymbol{u}^{(1)} \cdots \boldsymbol{u}^{(i)}$.

Now observe that for every $i \in [\lambda]$, $\mathcal{D}_{i-1}$ and $\mathcal{D}_i$ are ϵ-indistin guishable following immediately from the ϵ-indistinguishability of the δ-$\mathsf{LPN}_{\mathbb{F}}$ assumption. Finally, observe that $\mathcal{D}_\lambda$ and $\mathcal{D}_{\lambda+1}$ are ϵ-indistinguishable due to the ϵ-security of the PRG $(\mathsf{IdSamp}, \mathsf{Eval})$. Therefore, the lemma holds. □

□

5 BOOTSTRAPPING TO INDISTINGUISHABILITY OBFUSCATION

A sequence of prior works led to the work of [68] that constructs *iO* from a structured-seed PRG, and the bilateral-DLIN assumption and LWE assumptions. We first recall their theorem (See Theorem 10.2 in [68]):

Theorem 5.1 ([68]). *Let* τ *be arbitrary constant greater than 0,* ϵ *in* $(0,1)$ *and* $d \in \mathbb{N}$ *be any constants. Assume sub-exponential security of the following assumptions, where* λ *is the security parameter,* p *is a* λ*-bit prime, and the parameter* k *is a large enough polynomial in* λ*:*

- *Bilateral-*DLIN *assumption on asymmetric bilinear groups of order* p *(implied by the* DLIN *assumption on symmetric bilinear groups of order* p*),*
- LWE *assumption over* $\mathbb{Z}_p$ *with subexponential modulus-to-noise ratio* 2^{k^ϵ}*, where* k *is the dimension of the* LWE *secret.*
- *Existence of a structured-seed PRG with stretch* $1+\tau$ *and complexity* $(\deg d, \deg 2)$ *over* $\mathbb{Z}_p$ *as in Definition 3.1.*

Then, there exists a (subexponentially secure) indistinguishability obfuscation scheme for all circuits. Further if these assumptions are polynomially secure, then there exists a compact functional encryption scheme for all circuits.

Combining the Theorem above with Theorem 4.1, where we construct a structured seed PRG, we get the the following theorem.

Theorem 5.2. *Let* τ, ϵ *be arbitrary constants greater than 0, and* δ *in* $(0,1)$ *be a constant. Assume sub-exponential security of the following assumptions,*

- *the* LWE *assumption with subexponential modulus-to-noise ratio* 2^{k^ϵ} *and noises of magnitude polynomial in* k*, where* k *is the dimension of the* LWE *secret,*
- *the* δ-$\mathsf{LPN}_{\mathbb{F}}$ *assumption (as in Definition 2.3),*
- *the existence of a Boolean* PRG *in* NC^0 *with stretch* $1+\tau$*,*
- *Bilateral-*DLIN *assumption on asymmetric bilinear groups of prime order (implied by the* DLIN *assumption on symmetric bilinear groups of prime order),*

Then, (subexponentially secure) indistinguishability obfuscation for all polynomial-size circuits exists. Further if these assumptions are polynomially secure, then there exists a compact functional encryption scheme for all circuits.

ACKNOWLEDGEMENTS

We would like to thank Stefano Tessaro for helpful discussions. We thank Brent Waters for very detailed feedback on the initial public posting of this work. We thank James Bartusek, Geoffroy Couteau and Yuval Ishai for insightful discussions about our assumptions. In particular, we thank Geoffroy Couteau for providing very detailed background on the landscape of attacks on the LPN assumption over fields. We thank the anonymous reviewers for their comments and suggestions. We would also like to thank the Simons Institute for the Theory of Computing, for hosting all three authors during the program entitled "Lattices: Algorithms, Complexity, and Cryptography".

Aayush Jain was partially supported by grants listed under Amit Sahai, a Google PhD fellowship and a DIMACS award. This work

was partly carried out while the author was an intern at NTT Research. This work was partly carried out during a research visit conducted with support from DIMACS in association with its Special Focus on Cryptography.

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, a Simons collaboration grant on algorithmic fairness, the Defense Advanced Research Projects Agency (DARPA) and Army Research Office (ARO) under 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, an equipment grant from Intel, and an Okawa Foundation Research Grant. This material is based upon work supported by the Defense Advanced Research Projects Agency through Award HR00112020024 and the ARL under Contract W911NF-15-C- 0205. Amit Sahai is also grateful for the contributions of the LADWP to this effort.

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, Simons foundation, the National Science Foundation, NTT Research, Google, or the U.S. Government.

REFERENCES

[1] Tim Abbot, Daniel Kane, and Paul Valiant. 2004. On algorithms for nash equilibria. unpublished manuscript. https://web.mit.edu/tabbott/Public/final.pdf.

[2] Navid Alamati, Hart Montgomery, and Sikhar Patranabis. 2019. Symmetric Primitives with Structured Secrets. Cryptology ePrint Archive, Report 2019/608. https://eprint.iacr.org/2019/608.

[3] Martin R. Albrecht, Pooya Farshim, Dennis Hofheinz, Enrique Larraia, and Kenneth G. Paterson. 2016. Multilinear Maps from Obfuscation. In *TCC 2016-A, Part I (LNCS, Vol. 9562)*, Eyal Kushilevitz and Tal Malkin (Eds.). Springer, Heidelberg, 446–473. https://doi.org/10.1007/978-3-662-49096-9_19

[4] Michael Alekhnovich. 2003. More on Average Case vs Approximation Complexity. In *44th FOCS*. IEEE Computer Society Press, 298–307. https://doi.org/10.1109/SFCS.2003.1238204

[5] Prabhanjan Ananth, Yu-Chi Chen, Kai-Min Chung, Huijia Lin, and Wei-Kai Lin. 2016. Delegating RAM Computations with Adaptive Soundness and Privacy. In *TCC 2016-B, Part II (LNCS, Vol. 9986)*, Martin Hirt and Adam D. Smith (Eds.). Springer, Heidelberg, 3–30. https://doi.org/10.1007/978-3-662-53644-5_1

[6] Prabhanjan Ananth, Aayush Jain, Huijia Lin, Christian Matt, and Amit Sahai. 2019. Indistinguishability Obfuscation Without Multilinear Maps: New Paradigms via Low Degree Weak Pseudorandomness and Security Amplification. In *CRYPTO 2019, Part III (LNCS, Vol. 11694)*, Alexandra Boldyreva and Daniele Micciancio (Eds.). Springer, Heidelberg, 284–332. https://doi.org/10.1007/978-3-030-26954-8_10

[7] Prabhanjan Ananth, Aayush Jain, and Amit Sahai. 2018. Indistinguishability Obfuscation Without Multilinear Maps: iO from LWE, Bilinear Maps, and Weak Pseudorandomness. *IACR Cryptology ePrint Archive* 2018 (2018), 615.

[8] Prabhanjan Ananth and Alex Lombardi. 2018. Succinct Garbling Schemes from Functional Encryption Through a Local Simulation Paradigm. In *TCC 2018, Part II (LNCS, Vol. 11240)*, Amos Beimel and Stefan Dziembowski (Eds.). Springer, Heidelberg, 455–472. https://doi.org/10.1007/978-3-030-03810-6_17

[9] Benny Applebaum. 2012. Pseudorandom generators with long stretch and low locality from random local one-way functions. In *44th ACM STOC*, Howard J. Karloff and Toniann Pitassi (Eds.). ACM Press, 805–816. https://doi.org/10.1145/2213977.2214050

[10] Benny Applebaum. 2013. Pseudorandom Generators with Long Stretch and Low Locality from Random Local One-Way Functions. *SIAM J. Comput.* 42, 5 (2013), 2008–2037. https://doi.org/10.1137/120884857

[11] Benny Applebaum, Jonathan Avron, and Christina Brzuska. 2015. Arithmetic Cryptography: Extended Abstract. In *ITCS 2015*, Tim Roughgarden (Ed.). ACM, 143–151. https://doi.org/10.1145/2688073.2688114

[12] Benny Applebaum, Boaz Barak, and Avi Wigderson. 2010. Public-key cryptography from different assumptions. In *42nd ACM STOC*, Leonard J. Schulman (Ed.). ACM Press, 171–180. https://doi.org/10.1145/1806689.1806714

[13] Benny Applebaum, Andrej Bogdanov, and Alon Rosen. 2012. A Dichotomy for Local Small-Bias Generators. In *TCC 2012 (LNCS, Vol. 7194)*, Ronald Cramer (Ed.). Springer, Heidelberg, 600–617. https://doi.org/10.1007/978-3-642-28914-9_34

[14] Benny Applebaum, Ivan Damgård, Yuval Ishai, Michael Nielsen, and Lior Zichron. 2017. Secure Arithmetic Computation with Constant Computational Overhead. In *CRYPTO 2017, Part I (LNCS, Vol. 10401)*, Jonathan Katz and Hovav Shacham (Eds.). Springer, Heidelberg, 223–254. https://doi.org/10.1007/978-3-319-63688-7_8

[15] Benny Applebaum and Eliran Kachlon. 2019. Sampling Graphs without Forbidden Subgraphs and Unbalanced Expanders with Negligible Error. In *60th FOCS*, David Zuckerman (Ed.). IEEE Computer Society Press, 171–179. https://doi.org/10.1109/FOCS.2019.00020

[16] Benny Applebaum and Shachar Lovett. 2016. Algebraic attacks against random local functions and their countermeasures. In *48th ACM STOC*, Daniel Wichs and Yishay Mansour (Eds.). ACM Press, 1087–1100. https://doi.org/10.1145/2897518.2897554

[17] Saikrishna Badrinarayanan, Vipul Goyal, Aayush Jain, and Amit Sahai. 2016. Verifiable Functional Encryption. In *ASIACRYPT 2016, Part II (LNCS, Vol. 10032)*, Jung Hee Cheon and Tsuyoshi Takagi (Eds.). Springer, Heidelberg, 557–587. https://doi.org/10.1007/978-3-662-53890-6_19

[18] Boaz Barak, Zvika Brakerski, Ilan Komargodski, and Pravesh Kothari. 2017. Limits on Low-Degree Pseudorandom Generators (Or: Sum-of-Squares Meets Program Obfuscation). *Electronic Colloquium on Computational Complexity (ECCC)* 24 (2017), 60.

[19] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. 2001. On the (Im)possibility of Obfuscating Programs. In *CRYPTO 2001 (LNCS, Vol. 2139)*, Joe Kilian (Ed.). Springer, Heidelberg, 1–18. https://doi.org/10.1007/3-540-44647-8_1

[20] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. 2001. On the (Im)possibility of Obfuscating Programs. In *Advances in Cryptology - CRYPTO 2001, 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19-23, 2001, Proceedings*. 1–18.

[21] Boaz Barak, Samuel B. Hopkins, Aayush Jain, Pravesh Kothari, and Amit Sahai. 2019. Sum-of-Squares Meets Program Obfuscation, Revisited. In *EUROCRYPT 2019, Part I (LNCS, Vol. 11476)*, Yuval Ishai and Vincent Rijmen (Eds.). Springer, Heidelberg, 226–250. https://doi.org/10.1007/978-3-030-17653-2_8

[22] James Bartusek, Tancrède Lepoint, Fermi Ma, and Mark Zhandry. 2019. New Techniques for Obfuscating Conjunctions. In *EUROCRYPT 2019, Part III (LNCS, Vol. 11478)*, Yuval Ishai and Vincent Rijmen (Eds.). Springer, Heidelberg, 636–666. https://doi.org/10.1007/978-3-030-17659-4_22

[23] Allison Bishop, Abhishek Jain, and Lucas Kowalczyk. 2015. Function-Hiding Inner Product Encryption. In *ASIACRYPT 2015, Part I (LNCS, Vol. 9452)*, Tetsu Iwata and Jung Hee Cheon (Eds.). Springer, Heidelberg, 470–491. https://doi.org/10.1007/978-3-662-48797-6_20

[24] Allison Bishop, Lucas Kowalczyk, Tal Malkin, Valerio Pastro, Mariana Raykova, and Kevin Shi. 2019. In Pursuit of Clarity In Obfuscation. *IACR Cryptol. ePrint Arch.* 2019 (2019), 463. https://eprint.iacr.org/2019/463

[25] Nir Bitansky, Sanjam Garg, Huijia Lin, Rafael Pass, and Sidharth Telang. 2015. Succinct Randomized Encodings and their Applications. In *47th ACM STOC*, Rocco A. Servedio and Ronitt Rubinfeld (Eds.). ACM Press, 439–448. https://doi.org/10.1145/2746539.2746574

[26] Nir Bitansky, Omer Paneth, and Alon Rosen. 2015. On the Cryptographic Hardness of Finding a Nash Equilibrium. In *56th FOCS*, Venkatesan Guruswami (Ed.). IEEE Computer Society Press, 1480–1498. https://doi.org/10.1109/FOCS.2015.94

[27] Avrim Blum, Merrick L. Furst, Michael J. Kearns, and Richard J. Lipton. 1994. Cryptographic Primitives Based on Hard Learning Problems. In *CRYPTO'93 (LNCS, Vol. 773)*, Douglas R. Stinson (Ed.). Springer, Heidelberg, 278–291. https://doi.org/10.1007/3-540-48329-2_24

[28] Andrej Bogdanov and Youming Qiao. 2009. On the Security of Goldreich's One-Way Function. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, 12th International Workshop, APPROX 2009, and 13th International Workshop, RANDOM 2009, Berkeley, CA, USA, August 21-23, 2009. Proceedings (Lecture Notes in Computer Science, Vol. 5687)*, Irit Dinur, Klaus Jansen, Joseph Naor, and José D. P. Rolim (Eds.). Springer, 392–405. https://doi.org/10.1007/978-3-642-03685-9_30

[29] Andrej Bogdanov and Youming Qiao. 2012. On the security of Goldreich's one-way function. *Comput. Complex.* 21, 1 (2012), 83–127.

[30] Dan Boneh, Xavier Boyen, and Hovav Shacham. 2004. Short Group Signatures. In *CRYPTO 2004 (LNCS, Vol. 3152)*, Matthew Franklin (Ed.). Springer, Heidelberg, 41–55. https://doi.org/10.1007/978-3-540-28628-8_3

[31] Dan Boneh, Rosario Gennaro, Steven Goldfeder, Aayush Jain, Sam Kim, Peter M. R. Rasmussen, and Amit Sahai. 2018. Threshold Cryptosystems from Threshold Fully Homomorphic Encryption. In *Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part I (Lecture Notes in Computer Science, Vol. 10991)*,

Hovav Shacham and Alexandra Boldyreva (Eds.). Springer, 565–596.
[32] Dan Boneh, Craig Gentry, Sergey Gorbunov, Shai Halevi, Valeria Nikolaenko, Gil Segev, Vinod Vaikuntanathan, and Dhinakaran Vinayagamurthy. 2014. Fully Key-Homomorphic Encryption, Arithmetic Circuit ABE and Compact Garbled Circuits. In *Advances in Cryptology - EUROCRYPT 2014 - 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings (Lecture Notes in Computer Science, Vol. 8441)*, Phong Q. Nguyen and Elisabeth Oswald (Eds.). Springer, 533–556. https://doi.org/10.1007/978-3-642-55220-5_30
[33] Dan Boneh, David J. Wu, and Joe Zimmerman. 2014. Immunizing Multilinear Maps Against Zeroizing Attacks. Cryptology ePrint Archive, Report 2014/930.
[34] Dan Boneh and Mark Zhandry. 2014. Multiparty Key Exchange, Efficient Traitor Tracing, and More from Indistinguishability Obfuscation. In *CRYPTO 2014, Part I (LNCS, Vol. 8616)*, Juan A. Garay and Rosario Gennaro (Eds.). Springer, Heidelberg, 480–499. https://doi.org/10.1007/978-3-662-44371-2_27
[35] Elette Boyle, Geoffroy Couteau, Niv Gilboa, and Yuval Ishai. 2018. Compressing Vector OLE. In *ACM CCS 2018*, David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang (Eds.). ACM Press, 896–912. https://doi.org/10.1145/3243734.3243868
[36] Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, Peter Rindal, and Peter Scholl. 2019. Efficient Two-Round OT Extension and Silent Non-Interactive Secure Computation. In *ACM CCS 2019*, Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz (Eds.). ACM Press, 291–308. https://doi.org/10.1145/3319535.3354255
[37] Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, and Peter Scholl. 2020. Correlated Pseudorandom Functions from Variable-Density LPN. In *61st FOCS*. IEEE Computer Society Press, 1069–1080. https://doi.org/10.1109/FOCS46700.2020.00103
[38] Zvika Brakerski, Nico Döttling, Sanjam Garg, and Giulio Malavolta. 2020. Candidate iO from Homomorphic Encryption Schemes. Cryptology ePrint Archive, Report 2020/394. https://eprint.iacr.org/2020/394.
[39] Zvika Brakerski, Nico Döttling, Sanjam Garg, and Giulio Malavolta. 2020. Candidate iO from Homomorphic Encryption Schemes. In *EUROCRYPT 2020, Part I (LNCS)*, Vincent Rijmen and Yuval Ishai (Eds.). Springer, Heidelberg, 79–109. https://doi.org/10.1007/978-3-030-45721-1_4
[40] Zvika Brakerski, Craig Gentry, Shai Halevi, Tancrede Lepoint, Amit Sahai, and Mehdi Tibouchi. 2015. Cryptanalysis of the Quadratic Zero-Testing of GGH. Cryptology ePrint Archive, Report 2015/845. http://eprint.iacr.org/.
[41] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. 2012. (Leveled) fully homomorphic encryption without bootstrapping. In *ITCS 2012*, Shafi Goldwasser (Ed.). ACM, 309–325. https://doi.org/10.1145/2090236.2090262
[42] Zvika Brakerski, Yael Tauman Kalai, Jonathan Katz, and Vinod Vaikuntanathan. 2010. Overcoming the Hole in the Bucket: Public-Key Cryptography Resilient to Continual Memory Leakage. In *51st FOCS*. IEEE Computer Society Press, 501–510. https://doi.org/10.1109/FOCS.2010.55
[43] Zvika Brakerski and Vinod Vaikuntanathan. 2011. Efficient Fully Homomorphic Encryption from (Standard) LWE. In *52nd FOCS*, Rafail Ostrovsky (Ed.). IEEE Computer Society Press, 97–106. https://doi.org/10.1109/FOCS.2011.12
[44] Christina Brzuska, Pooya Farshim, and Arno Mittelbach. 2014. Indistinguishability Obfuscation and UCEs: The Case of Computationally Unpredictable Sources. In *CRYPTO 2014, Part I (LNCS, Vol. 8616)*, Juan A. Garay and Rosario Gennaro (Eds.). Springer, Heidelberg, 188–205. https://doi.org/10.1007/978-3-662-44371-2_11
[45] Ran Canetti, Yilei Chen, Justin Holmgren, Alex Lombardi, Guy N. Rothblum, Ron D. Rothblum, and Daniel Wichs. 2019. Fiat-Shamir: from practice to theory. In *51st ACM STOC*, Moses Charikar and Edith Cohen (Eds.). ACM Press, 1082–1090. https://doi.org/10.1145/3313276.3316380
[46] Ran Canetti, Yilei Chen, Justin Holmgren, and Mariana Raykova. 2016. Adaptive Succinct Garbled RAM or: How to Delegate Your Database. In *TCC 2016-B, Part II (LNCS, Vol. 9986)*, Martin Hirt and Adam D. Smith (Eds.). Springer, Heidelberg, 61–90. https://doi.org/10.1007/978-3-662-53644-5_3
[47] Ran Canetti, Yilei Chen, and Leonid Reyzin. 2016. On the Correlation Intractability of Obfuscated Pseudorandom Functions. In *TCC 2016-A, Part I (LNCS, Vol. 9562)*, Eyal Kushilevitz and Tal Malkin (Eds.). Springer, Heidelberg, 389–415. https://doi.org/10.1007/978-3-662-49096-9_17
[48] Ran Canetti and Justin Holmgren. 2016. Fully Succinct Garbled RAM. In *ITCS 2016*, Madhu Sudan (Ed.). ACM, 169–178. https://doi.org/10.1145/2840728.2840765
[49] Ran Canetti, Justin Holmgren, Abhishek Jain, and Vinod Vaikuntanathan. 2015. Succinct Garbling and Indistinguishability Obfuscation for RAM Programs. In *47th ACM STOC*, Rocco A. Servedio and Ronitt Rubinfeld (Eds.). ACM Press, 429–437. https://doi.org/10.1145/2746539.2746621
[50] Ran Canetti, Huijia Lin, Stefano Tessaro, and Vinod Vaikuntanathan. 2015. Obfuscation of Probabilistic Circuits and Applications. In *TCC 2015, Part II (LNCS, Vol. 9015)*, Yevgeniy Dodis and Jesper Buus Nielsen (Eds.). Springer, Heidelberg, 468–497. https://doi.org/10.1007/978-3-662-46497-7_19
[51] Ran Canetti, Sunoo Park, and Oxana Poburinnaya. 2020. Fully Deniable Interactive Encryption. In *CRYPTO 2020, Part I (LNCS)*, Hovav Shacham and Alexandra Boldyreva (Eds.). Springer, Heidelberg, 807–835. https://doi.org/10.1007/978-3-030-56784-2_27
[52] Yu-Chi Chen, Sherman S. M. Chow, Kai-Min Chung, Russell W. F. Lai, Wei-Kai Lin, and Hong-Sheng Zhou. 2015. Cryptography for Parallel RAM from Indistinguishability Obfuscation. Cryptology ePrint Archive, Report 2015/406. http://eprint.iacr.org/2015/406.
[53] Jung Hee Cheon, Kyoohyung Han, Changmin Lee, Hansol Ryu, and Damien Stehlé. 2015. Cryptanalysis of the Multilinear Map over the Integers. In *EUROCRYPT*.
[54] Jung Hee Cheon, Changmin Lee, and Hansol Ryu. 2015. Cryptanalysis of the New CLT Multilinear Maps. Cryptology ePrint Archive, Report 2015/934. http://eprint.iacr.org/.
[55] Arka Rai Choudhuri, Pavel Hubácek, Chethan Kamath, Krzysztof Pietrzak, Alon Rosen, and Guy N. Rothblum. 2019. Finding a Nash equilibrium is no easier than breaking Fiat-Shamir. In *51st ACM STOC*, Moses Charikar and Edith Cohen (Eds.). ACM Press, 1103–1114. https://doi.org/10.1145/3313276.3316400
[56] Arka Rai Choudhuri, Pavel Hubacek, Chethan Kamath, Krzysztof Pietrzak, Alon Rosen, and Guy N. Rothblum. 2019. PPAD-Hardness via Iterated Squaring Modulo a Composite. Cryptology ePrint Archive, Report 2019/667. https://eprint.iacr.org/2019/667.
[57] Kai-Min Chung, Huijia Lin, and Rafael Pass. 2015. Constant-Round Concurrent Zero-Knowledge from Indistinguishability Obfuscation. In *CRYPTO 2015, Part I (LNCS, Vol. 9215)*, Rosario Gennaro and Matthew J. B. Robshaw (Eds.). Springer, Heidelberg, 287–307. https://doi.org/10.1007/978-3-662-47989-6_14
[58] Aloni Cohen, Justin Holmgren, Ryo Nishimaki, Vinod Vaikuntanathan, and Daniel Wichs. 2016. Watermarking cryptographic capabilities. In *STOC*.
[59] James Cook, Omid Etesami, Rachel Miller, and Luca Trevisan. 2009. Goldreich's One-Way Function Candidate and Myopic Backtracking Algorithms. In *TCC 2009 (LNCS, Vol. 5444)*, Omer Reingold (Ed.). Springer, Heidelberg, 521–538. https://doi.org/10.1007/978-3-642-00457-5_31
[60] Jean-Sébastien Coron, Craig Gentry, Shai Halevi, Tancrède Lepoint, Hemanta K. Maji, Eric Miles, Mariana Raykova, Amit Sahai, and Mehdi Tibouchi. 2015. Zeroizing Without Low-Level Zeroes: New MMAP Attacks and Their Limitations. In *CRYPTO*.
[61] Geoffroy Couteau, Aurélien Dupin, Pierrick Méaux, Mélissa Rossi, and Yann Rotella. 2018. On the Concrete Security of Goldreich's Pseudorandom Generator. In *ASIACRYPT 2018, Part II (LNCS, Vol. 11273)*, Thomas Peyrin and Steven Galbraith (Eds.). Springer, Heidelberg, 96–124. https://doi.org/10.1007/978-3-030-03329-3_4
[62] Mary Cryan and Peter Bro Miltersen. 2001. On Pseudorandom Generators in NC. In *Mathematical Foundations of Computer Science 2001, 26th International Symposium, MFCS 2001 Marianske Lazne, Czech Republic, August 27-31, 2001, Proceedings (Lecture Notes in Computer Science, Vol. 2136)*, Jiri Sgall, Ales Pultr, and Petr Kolman (Eds.). Springer, 272–284.
[63] Nico Döttling, Satrajit Ghosh, Jesper Buus Nielsen, Tobias Nilges, and Roberto Trifiletti. 2017. TinyOLE: Efficient Actively Secure Two-Party Computation from Oblivious Linear Function Evaluation. In *ACM CCS 2017*, Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu (Eds.). ACM Press, 2263–2276. https://doi.org/10.1145/3133956.3134024
[64] Naomi Ephraim, Cody Freitag, Ilan Komargodski, and Rafael Pass. 2020. Continuous Verifiable Delay Functions. In *EUROCRYPT 2020, Part III (LNCS)*, Vincent Rijmen and Yuval Ishai (Eds.). Springer, Heidelberg, 125–154. https://doi.org/10.1007/978-3-030-45727-3_5
[65] Pooya Farshim, Julia Hesse, Dennis Hofheinz, and Enrique Larraia. 2018. Graded Encoding Schemes from Obfuscation. In *PKC 2018, Part II (LNCS, Vol. 10770)*, Michel Abdalla and Ricardo Dahab (Eds.). Springer, Heidelberg, 371–400. https://doi.org/10.1007/978-3-319-76581-5_13
[66] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. 2013. Candidate Indistinguishability Obfuscation and Functional Encryption for all Circuits. In *54th FOCS*. IEEE Computer Society Press, 40–49. https://doi.org/10.1109/FOCS.2013.13
[67] Sanjam Garg, Omkant Pandey, and Akshayaram Srinivasan. 2016. Revisiting the Cryptographic Hardness of Finding a Nash Equilibrium. In *CRYPTO 2016, Part II (LNCS, Vol. 9815)*, Matthew Robshaw and Jonathan Katz (Eds.). Springer, Heidelberg, 579–604. https://doi.org/10.1007/978-3-662-53008-5_20
[68] Romain Gay, Aayush Jain, Huijia Lin, and Amit Sahai. 2020. Indistinguishability Obfuscation from Simple-to-State Hard Problems: New Assumptions, New Techniques, and Simplification. *IACR Cryptol. ePrint Arch.* 2020 (2020), 764.
[69] Romain Gay and Rafael Pass. 2020. Indistinguishability Obfuscation from Circular Security. Cryptology ePrint Archive, Report 2020/1010. https://eprint.iacr.org/2020/1010.
[70] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. 2008. Trapdoors for hard lattices and new cryptographic constructions. In *40th ACM STOC*, Richard E. Ladner and Cynthia Dwork (Eds.). ACM Press, 197–206. https://doi.org/10.1145/1374376.1374407
[71] Craig Gentry, Amit Sahai, and Brent Waters. 2013. Homomorphic Encryption from Learning with Errors: Conceptually-Simpler, Asymptotically-Faster, Attribute-Based. In *CRYPTO 2013, Part I (LNCS, Vol. 8042)*, Ran Canetti and

Juan A. Garay (Eds.). Springer, Heidelberg, 75–92. https://doi.org/10.1007/978-3-642-40041-4_5

[72] Craig Gentry and Daniel Wichs. 2011. Separating succinct non-interactive arguments from all falsifiable assumptions. In *43rd ACM STOC*, Lance Fortnow and Salil P. Vadhan (Eds.). ACM Press, 99–108. https://doi.org/10.1145/1993636.1993651

[73] Satrajit Ghosh, Jesper Buus Nielsen, and Tobias Nilges. 2017. Maliciously Secure Oblivious Linear Function Evaluation with Constant Overhead. In *ASIACRYPT 2017, Part I (LNCS, Vol. 10624)*, Tsuyoshi Takagi and Thomas Peyrin (Eds.). Springer, Heidelberg, 629–659. https://doi.org/10.1007/978-3-319-70694-8_22

[74] E. N. Gilbert. 1952. A comparison of signalling alphabets. *The Bell System Technical Journal* 31, 3 (1952), 504–522.

[75] Oded Goldreich. 2000. Candidate One-Way Functions Based on Expander Graphs. *Electronic Colloquium on Computational Complexity (ECCC)* 7, 90 (2000).

[76] Shafi Goldwasser, S. Dov Gordon, Vipul Goyal, Abhishek Jain, Jonathan Katz, Feng-Hao Liu, Amit Sahai, Elaine Shi, and Hong-Sheng Zhou. 2014. Multi-input Functional Encryption. In *EUROCRYPT 2014 (LNCS, Vol. 8441)*, Phong Q. Nguyen and Elisabeth Oswald (Eds.). Springer, Heidelberg, 578–602. https://doi.org/10.1007/978-3-642-55220-5_32

[77] Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. 2008. One-Time Programs. In *CRYPTO 2008 (LNCS, Vol. 5157)*, David Wagner (Ed.). Springer, Heidelberg, 39–56. https://doi.org/10.1007/978-3-540-85174-5_3

[78] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. 2013. Attribute-based encryption for circuits. In *45th ACM STOC*, Dan Boneh, Tim Roughgarden, and Joan Feigenbaum (Eds.). ACM Press, 545–554. https://doi.org/10.1145/2488608.2488677

[79] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. 2015. Predicate Encryption for Circuits from LWE. In *CRYPTO 2015, Part II (LNCS, Vol. 9216)*, Rosario Gennaro and Matthew J. B. Robshaw (Eds.). Springer, Heidelberg, 503–523. https://doi.org/10.1007/978-3-662-48000-7_25

[80] Rishab Goyal, Venkata Koppula, and Brent Waters. 2017. Lockable Obfuscation. In *58th FOCS*, Chris Umans (Ed.). IEEE Computer Society Press, 612–621. https://doi.org/10.1109/FOCS.2017.62

[81] Jens Groth and Amit Sahai. 2008. Efficient Non-interactive Proof Systems for Bilinear Groups. In *EUROCRYPT 2008 (LNCS, Vol. 4965)*, Nigel P. Smart (Ed.). Springer, Heidelberg, 415–432. https://doi.org/10.1007/978-3-540-78967-3_24

[82] Shai Halevi. 2015. Graded Encoding, Variations on a Scheme. *IACR Cryptology ePrint Archive* 2015 (2015), 866.

[83] Dennis Hofheinz, Tibor Jager, Dakshita Khurana, Amit Sahai, Brent Waters, and Mark Zhandry. 2016. How to Generate and Use Universal Samplers. In *ASIACRYPT 2016, Part II (LNCS, Vol. 10032)*, Jung Hee Cheon and Tsuyoshi Takagi (Eds.). Springer, Heidelberg, 715–744. https://doi.org/10.1007/978-3-662-53890-6_24

[84] Susan Hohenberger, Amit Sahai, and Brent Waters. 2013. Full Domain Hash from (Leveled) Multilinear Maps and Identity-Based Aggregate Signatures. In *CRYPTO 2013, Part I (LNCS, Vol. 8042)*, Ran Canetti and Juan A. Garay (Eds.). Springer, Heidelberg, 494–512. https://doi.org/10.1007/978-3-642-40041-4_27

[85] Yupu Hu and Huiwen Jia. 2015. Cryptanalysis of GGH Map. *IACR Cryptology ePrint Archive* 2015 (2015), 301.

[86] Pavel Hubácek and Eylon Yogev. 2017. Hardness of Continuous Local Search: Query Complexity and Cryptographic Lower Bounds. In *28th SODA*, Philip N. Klein (Ed.). ACM-SIAM, 1352–1371. https://doi.org/10.1137/1.9781611974782.88

[87] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. 2008. Cryptography with constant computational overhead. In *40th ACM STOC*, Richard E. Ladner and Cynthia Dwork (Eds.). ACM Press, 433–442. https://doi.org/10.1145/1374376.1374438

[88] Yuval Ishai, Manoj Prabhakaran, and Amit Sahai. 2009. Secure Arithmetic Computation with No Honest Majority. In *TCC Conference, TCC 2009, San Francisco, CA, USA, March 15-17, 2009. Proceedings*. 294–314.

[89] Aayush Jain, Huijia Lin, Christian Matt, and Amit Sahai. 2019. How to Leverage Hardness of Constant-Degree Expanding Polynomials overa $\mathbb{R}$ to build $i\mathcal{O}$. In *EUROCRYPT 2019, Part I (LNCS, Vol. 11476)*, Yuval Ishai and Vincent Rijmen (Eds.). Springer, Heidelberg, 251–281. https://doi.org/10.1007/978-3-030-17653-2_9

[90] Aayush Jain, Huijia Lin, and Amit Sahai. 2019. Simplifying Constructions and Assumptions for $i\mathcal{O}$. *IACR Cryptol. ePrint Arch.* 2019 (2019), 1252. https://eprint.iacr.org/2019/1252

[91] Aayush Jain, Huijia Lin, and Amit Sahai. 2020. Indistinguishability Obfuscation from Well-Founded Assumptions. Cryptology ePrint Archive, Report 2020/1003. https://eprint.iacr.org/2020/1003.

[92] Ruta Jawale, Yael Tauman Kalai, Dakshita Khurana, and Rachel Zhang. 2020. SNARGs for Bounded Depth Computations and PPAD Hardness from Sub-Exponential LWE. Cryptology ePrint Archive, Report 2020/980. https://eprint.iacr.org/2020/980.

[93] Charanjit S. Jutla and Arnab Roy. 2013. Shorter Quasi-Adaptive NIZK Proofs for Linear Subspaces. In *ASIACRYPT 2013, Part I (LNCS, Vol. 8269)*, Kazue Sako and Palash Sarkar (Eds.). Springer, Heidelberg, 1–20. https://doi.org/10.1007/978-3-642-42033-7_1

[94] Yael Tauman Kalai, Guy N. Rothblum, and Ron D. Rothblum. 2017. From Obfuscation to the Security of Fiat-Shamir for Proofs. In *CRYPTO 2017, Part II (LNCS, Vol. 10402)*, Jonathan Katz and Hovav Shacham (Eds.). Springer, Heidelberg, 224–251. https://doi.org/10.1007/978-3-319-63715-0_8

[95] Dakshita Khurana, Vanishree Rao, and Amit Sahai. 2015. Multi-party Key Exchange for Unbounded Parties from Indistinguishability Obfuscation. In *ASIACRYPT 2015, Part I (LNCS, Vol. 9452)*, Tetsu Iwata and Jung Hee Cheon (Eds.). Springer, Heidelberg, 52–75. https://doi.org/10.1007/978-3-662-48797-6_3

[96] Ilan Komargodski, Tal Moran, Moni Naor, Rafael Pass, Alon Rosen, and Eylon Yogev. 2014. One-Way Functions and (Im)Perfect Obfuscation. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*. IEEE Computer Society, 374–383. https://doi.org/10.1109/FOCS.2014.47

[97] Ilan Komargodski, Moni Naor, and Eylon Yogev. 2014. Secret-Sharing for NP. In *ASIACRYPT 2014, Part II (LNCS, Vol. 8874)*, Palash Sarkar and Tetsu Iwata (Eds.). Springer, Heidelberg, 254–273. https://doi.org/10.1007/978-3-662-45608-8_14

[98] Ilan Komargodski and Gil Segev. 2017. From Minicrypt to Obfustopia via Private-Key Functional Encryption. In *EUROCRYPT 2017, Part I (LNCS, Vol. 10210)*, Jean-Sébastien Coron and Jesper Buus Nielsen (Eds.). Springer, Heidelberg, 122–151. https://doi.org/10.1007/978-3-319-56620-7_5

[99] Venkata Koppula, Allison Bishop Lewko, and Brent Waters. 2015. Indistinguishability Obfuscation for Turing Machines with Unbounded Memory. In *STOC*.

[100] Venkata Koppula, Allison Bishop Lewko, and Brent Waters. 2015. Indistinguishability Obfuscation for Turing Machines with Unbounded Memory. In *47th ACM STOC*, Rocco A. Servedio and Ronitt Rubinfeld (Eds.). ACM Press, 419–428. https://doi.org/10.1145/2746539.2746614

[101] Pravesh K. Kothari, Ryuhei Mori, Ryan O'Donnell, and David Witmer. 2017. Sum of squares lower bounds for refuting any CSP. In *49th ACM STOC*, Hamed Hatami, Pierre McKenzie, and Valerie King (Eds.). ACM Press, 132–145. https://doi.org/10.1145/3055399.3055485

[102] Huijia Lin and Christian Matt. 2018. Pseudo Flawed-Smudging Generators and Their Application to Indistinguishability Obfuscation. *IACR Cryptology ePrint Archive* 2018 (2018), 646.

[103] Alex Lombardi and Vinod Vaikuntanathan. 2017. Limits on the Locality of Pseudorandom Generators and Applications to Indistinguishability Obfuscation. In *TCC 2017, Part I (LNCS, Vol. 10677)*, Yael Kalai and Leonid Reyzin (Eds.). Springer, Heidelberg, 119–137. https://doi.org/10.1007/978-3-319-70500-2_5

[104] Alex Lombardi and Vinod Vaikuntanathan. 2020. Fiat-Shamir for Repeated Squaring with Applications to PPAD-Hardness and VDFs. In *CRYPTO 2020, Part III (LNCS)*, Hovav Shacham and Alexandra Boldyreva (Eds.). Springer, Heidelberg, 632–651. https://doi.org/10.1007/978-3-030-56877-1_22

[105] Eric Miles, Amit Sahai, and Mark Zhandry. 2016. Annihilation Attacks for Multilinear Maps: Cryptanalysis of Indistinguishability Obfuscation over GGH13. In *Advances in Cryptology - CRYPTO*.

[106] Brice Minaud and Pierre-Alain Fouque. 2015. Cryptanalysis of the New Multilinear Map over the Integers. Cryptology ePrint Archive, Report 2015/941. http://eprint.iacr.org/.

[107] Elchanan Mossel, Amir Shpilka, and Luca Trevisan. 2003. On e-Biased Generators in NC0. In *44th FOCS*. IEEE Computer Society Press, 136–145. https://doi.org/10.1109/SFCS.2003.1238188

[108] Ryan O'Donnell and David Witmer. 2014. 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*. IEEE Computer Society, 1–12. https://doi.org/10.1109/CCC.2014.9

[109] Tatsuaki Okamoto and Katsuyuki Takashima. 2010. Fully Secure Functional Encryption with General Relations from the Decisional Linear Assumption. In *CRYPTO 2010 (LNCS, Vol. 6223)*, Tal Rabin (Ed.). Springer, Heidelberg, 191–208. https://doi.org/10.1007/978-3-642-14623-7_11

[110] Oded Regev. 2005. On lattices, learning with errors, random linear codes, and cryptography. In *STOC*. 84–93.

[111] Amit Sahai and Brent Waters. 2014. How to use indistinguishability obfuscation: deniable encryption, and more. In *STOC*, David B. Shmoys (Ed.). ACM, 475–484. http://dl.acm.org/citation.cfm?id=2591796

[112] Rom Varshamov. 1957. Estimate of the number of signals in error correcting codes. *Dokl. Akad. Nauk SSSR* (1957).

[113] Hoeteck Wee and Daniel Wichs. 2020. Candidate Obfuscation via Oblivious LWE Sampling. *IACR Cryptol. ePrint Arch.* 2020 (2020), 1042. https://eprint.iacr.org/2020/1042

[114] Daniel Wichs and Giorgos Zirdelis. 2017. Obfuscating Compute-and-Compare Programs under LWE. In *58th FOCS*, Chris Umans (Ed.). IEEE Computer Society Press, 600–611. https://doi.org/10.1109/FOCS.2017.61

[115] Takashi Yamakawa, Shota Yamada, Goichiro Hanaoka, and Noboru Kunihiro. 2014. Self-bilinear Map on Unknown Order Groups from Indistinguishability Obfuscation and Its Applications. In *CRYPTO 2014, Part II (LNCS, Vol. 8617)*, Juan A. Garay and Rosario Gennaro (Eds.). Springer, Heidelberg, 90–107. https://doi.org/10.1007/978-3-662-44381-1_6