Electronic Colloquium on Computational Complexity, Report No. 126 (2020)

Indistinguishability Obfuscation from Well-Founded Assumptions

Aayush Jain[*] Huijia Lin[†] Amit Sahai[‡]

August 18, 2020

Abstract

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

Theorem (Informal)**.** Let $\tau \in (0, \infty), \delta \in (0, 1), \epsilon \in (0, 1)$ be arbitrary constants. Assume sub-exponential security of the following assumptions, where λ is a security parameter, and the parameters ℓ, k, n below are large enough polynomials in λ:

- the SXDH assumption on asymmetric bilinear groups of a prime order $p = O(2^\lambda)$,
- the LWE assumption over $\mathbb{Z}_p$ with subexponential modulus-to-noise ratio 2^{k^ϵ}, where k is the dimension of the LWE secret,
- the LPN assumption over $\mathbb{Z}_p$ 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 $n^{1+\tau}$,

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.

[*] UCLA, Center for Encrypted Functionalities, and NTT Research. Email: aayushjain@cs.ucla.edu.
[†] UW. Email: rachel@cs.washington.edu.
[‡] UCLA, Center for Encrypted Functionalities. Email: sahai@cs.ucla.edu.

ISSN 1433-8092

Contents

1 Introduction

In this work, we study the notion of indistinguishability obfuscation ($i\mathcal{O}$) for general polynomial-size circuits [BGI$^+$01a, GKR08, GGH$^+$13b]. $i\mathcal{O}$ requires that for any two circuits C_0 and C_1 of the same size, such that $\mathsf{C}_0(x) = \mathsf{C}_1(x)$ for all inputs x, we have that $i\mathcal{O}(\mathsf{C}_0)$ is computationally indistinguishable to $i\mathcal{O}(\mathsf{C}_1)$. 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. [GGH$^+$13b, SW14, BFM14, GGG$^+$14, HSW13, KLW15, BPR15, CHN$^+$16, GPS16, HJK$^+$16]).

Despite this success, until this work, all previously known $i\mathcal{O}$ constructions [GGH13a, GGH$^+$13b, BGK$^+$14, BR14, PST14, AGIS14, BMSZ16, CLT13, CLT15, GGH15, CHL$^+$15, BWZ14, CGH$^+$15, HJ15, BGH$^+$15, Hal15, CLR15, MF15, MSZ16, DGG$^+$16, Lin16, LV16, AS17, Lin17, LT17, GJK18, AJS18, Agr19, LM18, JLMS19, BIJ$^+$20, AP20, BDGM20] 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 [CHL$^+$15, BWZ14, CGH$^+$15, HJ15, BGH$^+$15, Hal15, CLR15, MF15, MSZ16, BBKK17, LV17, BHJ$^+$19]. 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 [BKM$^+$19]. 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, and the parameters ℓ, k, n below are large enough polynomials in λ:*

- *the* SXDH *assumption on asymmetric bilinear groups of a prime order $p = O(2^\lambda)$,*
- *the* LWE *assumption over $\mathbb{Z}_p$ with subexponential modulus-to-noise ratio 2^{k^ϵ}, where k is the dimension of the* LWE *secret,*
- *the* LPN *assumption over $\mathbb{Z}_p$ 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 $n^{1+\tau}$,*

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 $i\mathcal{O}$.

1.1 Assumptions in More Detail

We now describe each of these assumptions in more detail and briefly survey their history.

The SXDH Assumption: The standard SXDH assumption is stated as follows: Given an appropriate prime p, three groups $\mathbb{G}_1,\ \mathbb{G}_2$, and $\mathbb{G}_T$ are chosen of order p such that there exists an efficiently computable nontrivial bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. Canonical generators, g_1 for $\mathbb{G}_1$, and g_2 for $\mathbb{G}_1$, are also computed. Then, the SXDH assumption requires that the Decisional Diffie Hellman (DDH) assumption holds in both $\mathbb{G}_1$ and $\mathbb{G}_2$. That is, it requires that the following computational indistinguishability holds:

$$\forall b \in \{1,2\},\ \{(g_b^x, g_b^y, g_b^{xy}) \ \mid\ x, y \leftarrow \mathbb{Z}_p\} \approx_c \{(g_b^x, g_b^y, g_b^z) \ \mid\ x, y, z \leftarrow \mathbb{Z}_p\}$$

This assumption was first defined in the 2005 work of Ballard et. al. [BGdMM05]. Since then, SXDH has seen extensive use in a wide variety of applications throughout cryptography, including Identity-Based Encryption and Non-Interactive Zero Knowledge (See, e.g. [GS08, BKKV10, BJK15, Lin17, CLL$^+$12, JR13]). It has been a subject of extensive cryptanalytic study (see [Ver01] for early work and [GR04] for a survey).

The LWE Assumption: The LWE assumption with respect to subexponential-size modulus p, dimension λ, sample complexity $n(\lambda)$ and polynomial-expectation discrete Gaussian distribution χ over integers states that the following computational indistinguishability holds:

$$\{\mathbf{A}, \boldsymbol{s} \cdot \mathbf{A} + \boldsymbol{e} \mod p \ \mid \mathbf{A} \leftarrow \mathbb{Z}_p^{\lambda \times n},\ \boldsymbol{s} \leftarrow \mathbb{Z}_p^{1 \times \lambda},\ \boldsymbol{e} \leftarrow \chi^{1 \times n}\}$$
$$\approx_c\ \{\mathbf{A}, \boldsymbol{u}\ \mid \mathbf{A} \leftarrow \mathbb{Z}_p^{\lambda \times n},\ \boldsymbol{u} \leftarrow \mathbb{Z}_p^{1 \times n}\}$$

This assumption was first stated in the work of [Reg05]. The version stated above is provably hard as long as GAP-SVP. is hard to approximate to within subexponential factors in the worst case [Reg05, Pei09, GPV08, MR04, MP13]. LWE has been used extensively to construct applications such as Leveled Fully Homomorphic Encryption [BV11, BGV12, GSW13], Key-Homomorphic PRFs [BLMR13], Lockable Obfuscation [GKW17, WZ17], Homomorphic Secret-Sharing [MW16, DHRW16], Constrained PRFs [BV15b], Attribute Based Encryption [BGG$^+$14, GVW13, GVW15] and Universal Thresholdizers [BGG$^+$18], to name a few.

The existence of PRGs in NC^0: The assumption of the existence of a Boolean 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:

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

Pseudorandom generators are a fundamental primitive in their own right, and have vast applications throughout cryptography. PRGs in NC^0 are tightly connected to the fundamental topic of Constraint Satisfaction Problems (CSPs) in complexity theory, and were first proposed for cryptographic use by Goldreich [Gol00, CM01] 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 [Gol00, CM01, MST03, ABR12, BQ12, App12, OW14, AL16, KMOW17, CDM+18].

LPN **over large fields:** Like LWE, the 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 $n(\ell)$ and a noise rate $r = 1/\ell^\delta$ for $\delta \in (0,1)$ states that the following computational indistinguishability holds:

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

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 otherwise. 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 sparse random vector, but where it is nonzero, the entries have large expectation. The origins of the LPN assumption date all the way back to the 1950s: the works of Gilbert [Gil52] and Varshamov [Var57] 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 formally defined for general parameters in 2009 [IPS09], under the name "Assumption 2." While in [IPS09], 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 [Ale03] to construct public-key encryption with the field $\mathbb{F}_2$. The exact parameter settings that we describe above, with both general fields and polynomially low error, was explicitly posed by [BCGI18].

This assumption was posed for the purpose of building efficient secure two-party and multi-party protocols for arithmetic computations [IPS09, AAB15]. Earlier, LPN over binary fields was posed for the purpose of constructing identification schemes [HB01] and public-key encryption [Ale03]. Recently, the assumption has led to a wide variety of applications (see for example, [IPS09, AAB15, BCGI18, ADI+17, DGN+17, GNN17, BLMZ19, BCG+19]). A comprehensive review of known attacks on LPN over large fields, for the parameter settings we are interested in, was given in [BCGI18]. For our parameter setting, the best running time of known attacks is sub-exponential, for any choice of the constant $\delta \in (0,1)$ and for any polynomial $n(\ell)$

1.2 Our Ideas in a Nutshell

Previous work [AJS18, LM18, AJL+19, JLMS19, JLS19, GJLS20] showed that to achieve $i\mathcal{O}$, it is sufficient to assume LWE, SXDH, and PRG in NC^0, and one other object, that we will encapsulate as a *structured-seed* PRG (sPRG) with polynomial stretch and special efficiency properties. In an sPRG, the seed to the sPRG 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, say, 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:

$$\left\{(\boldsymbol{A},\ \boldsymbol{b} = \boldsymbol{s}\cdot\boldsymbol{A} + \boldsymbol{e} + \boldsymbol{\sigma},\ G(\boldsymbol{\sigma}))\ \mid \boldsymbol{A} \leftarrow \mathbb{Z}_p^{\ell\times n};\ \boldsymbol{s} \leftarrow \mathbb{Z}_p^{1\times\ell};\ \boldsymbol{e} \leftarrow \mathcal{D}_r^{1\times n}(p);\ \boldsymbol{\sigma} \leftarrow \{0,1\}^{1\times n}\right\}$$
$$\approx_c \left\{(\boldsymbol{A},\ \boldsymbol{u},\ \boldsymbol{w})\ \mid \boldsymbol{A} \leftarrow \mathbb{Z}_p^{\ell\times n};\ \boldsymbol{u} \leftarrow \mathbb{Z}_p^{1\times n};\ \boldsymbol{w} \leftarrow \{0,1\}^{1\times m(n)}\right\}$$

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 $\boldsymbol{e}$, by means of a simple pre-computation idea to "correct" for errors introduced due to this sparse error. A gentle overview is provided in Section 4, followed by our detailed construction and analysis.

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 the tensor product $\boldsymbol{v}^{\otimes k} = \underbrace{\boldsymbol{v} \otimes \cdots \otimes \boldsymbol{v}}_{k}$ to be the standard tensor product, but converted back into a vector. 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 $\mathrm{M} \in \mathbb{Z}_q^{n_1\times n_2}$, we denote by $|\mathrm{M}|$ the bit length of M. In this case, $|\mathrm{M}| = n_1 \cdot n_2 \cdot \log_2 q$. 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 $n, \lambda \in \mathbb{N}$. The intuition behind this definition is to think of n as being a sufficiently large polynomial in λ

Multilinear Representation of Polynomials and Representation over $\mathbb{Z}_p$. In this work we will consider multivariate polynomials $p \in \mathbb{Z}[\boldsymbol{x} = (x_1, \ldots, x_n)]$ mapping $\{0,1\}^n$ to $\{0,1\}$. For any such polynomial there is a unique multilinear polynomial p' (obtained by setting $x_i^2 = x_i$) such that $p' \in \mathbb{Z}[\boldsymbol{x}]$ and $p'(\boldsymbol{x}) = p(\boldsymbol{x})$ for all $\boldsymbol{x} \in \{0,1\}^n$. Further, such a polynomial can have a maximum degree of n. At times, we will consider polynomials $g \in \mathbb{Z}_p[\boldsymbol{x}]$ such that for every $\boldsymbol{x} \in \{0,1\}^n$, $g(\boldsymbol{x}) \mod p = p(\boldsymbol{x})$. Such a polynomial g can be constructed simply as follows. Let $p'(\boldsymbol{x}) = \sum_{S \subseteq [n]} c_S \ \Pi_{i \in S} x_i$. We can construct $g(\boldsymbol{x}) = \sum_{S \subseteq [n]} (c_S \mod p) \Pi_{i \in S} x_i$. Note that g has degree at most the degree of p' over $\mathbb{Z}$. For polynomials of degree d, both the process described above can take $O(n^d)$ time. In this work, we consider polynomials representing pseudorandom generators in NC^0. Such polynomials depend only on a constant number of input bits, and thus their multilinear representations (and their field representations) are also constant degree polynomials. In this scenario, these conversions take polynomial time.

Definition 2.1 ((T,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 (T,ϵ)-indistinguishable where $T : \mathbb{N} \to \mathbb{N}$ and $\epsilon : \mathbb{N} \to [0,1]$ if for every non-negative polynomial $\mathsf{poly}(\cdot,\cdot)$ and any adversary $\mathcal{A}$ running in time bounded by $T\,\mathsf{poly}(\lambda)$ it holds that: For every sufficiently large $\lambda \in \mathbb{N}$,*

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

We say that two ensembles are ϵ-indistinguishable if it is (λ,ϵ)-indistinguishable, and is subexponentially ϵ-indistinguishable if it is (T,ϵ)-indistinguishable for $T(\lambda) = 2^{\lambda^c}$ for some positive constant c. It is indistinguishable if it is $\frac{1}{\lambda^c}$-pseudorandom for every positive constant c, and subexponentially indistinguishable if $(T, 1/T)$-indistinguishable for $T(\lambda) = 2^{\lambda^c}$ for some positive constant c.

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

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. [BGI+01b], 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 (T,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*

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

- (T,γ)-**Indistinguishability:** *For every two ensembles $\{C_{0,\lambda}\}$ $\{C_{1,\lambda}\}$ 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 (T,γ)-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, [IPS09, AAB15, BCGI18, ADI+17, DGN+17, GNN17, BLMZ19, BCG+19]). We adopt the following definition from [BCGI18].

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.

Definition 2.3 ($\mathsf{LPN}(\ell,n,r,p)$-Assumption, [IPS09, AAB15, BCGI18])**.** *Let λ be the security parameter. For an efficiently computable prime modulus $p(\lambda)$, dimension $\ell(\lambda)$, sample complexity $n(\ell)$, and noise rate $r(n)$ we say that the $\mathsf{LPN}(\ell,n,r,p)$ assumption is (T,γ)-secure / hard / indistinguishable if the following two distributions are (T,γ)-indistinguishable:*

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

We will set ℓ to be a large enough polynomial in λ, set $r = \ell^{-\delta}$, for a constant $\delta \in (0,1)$, and set the number of samples $n = \ell^c$ for some constant $c > 1$. Note that this setting of parameters was considered in detail in the work of [BCGI18]. We refer the reader to [BCGI18] for a comprehensive discussion of the history and security of this assumption.

Leakage Lemma. We will use the following theorem in our security proofs.

Theorem 2.1 (Imported Theorem [CCL18])**.** *Let $n, \ell \in \mathbb{N}, \epsilon > 0$, and $\mathcal{C}_{leak}$ be a family of distinguisher circuits from $\{0,1\}^n \times \{0,1\}^\ell \rightarrow \{0,1\}$ of size $s(n)$. Then, for every distribution (X,W) over $\{0,1\}^n \times \{0,1\}^\ell$, there exists a simulator h such that:*

1. *h is computable by circuits of size bounded by $s' = O(s2^\ell\epsilon^{-2})$, and maps $\{0,1\}^n \times \{0,1\}^{s'} \rightarrow \{0,1\}^\ell$. We denote by U the uniform distribution over $\{0,1\}^{s'}$.*

2. *(X,W) and $(X, h(X,U))$ are ϵ-indistinguishable by $\mathcal{C}_{leak}$. That is, for every $C \in \mathcal{C}_{leak}$,*

$$\left| \Pr_{(x,w)\leftarrow(X,W)}[C(x,w)=1] - \Pr_{x\leftarrow X, u\leftarrow U}[C(x,h(x,u))=1] \right| \leq \epsilon$$

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 denote an sPRG to have 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*

$(T(\lambda), \gamma(\lambda))$**-pseudorandomness:** *the following distributions are (T, γ) indistinguishable.*

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

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,* $\mathsf{sPRG} \in (\mathbb{C},\ \textit{deg}\ d)$*, 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, 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* 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; the latter ensures that the computation on the public seed lies in $\mathsf{arith}\text{-}\mathsf{NC}_0$ (which is exactly the class of functions computed by constant-degree polynomials).

Theorem 4.1. *Let* λ *be the security parameter. Let* $d \in \mathbb{N}, \delta > 0, \tau > 1$ *be arbitrary constants and* $n = \mathsf{poly}(\lambda)$ *be an arbitrary positive non-constant polynomial.*

Then, assuming the following:

- *the existence of a constant locality Boolean* PRG *with stretch* $\tau > 1$ *and multilinear degree* d *over* $\mathbb{Z}$*, and,*
- $\mathsf{LPN}(\ell, n, r, p)$*-assumption holds with respect to dimension* $\ell = n^{1/\lceil \frac{d}{2} \rceil}$*, error rate* $r = \ell^{-\delta}$*,*

there exists an sPRG *with polynomial stretch in* $(\mathsf{arith}\text{-}\mathsf{NC}^0,\ \deg 2)$ *that is* γ*-pseudorandom for every constant* $\gamma > 0$*. Additionally, if both assumptions are secure against* 2^{λ^ν} *time adversaries for some constant* $\nu > 0$*, then,* sPRG *is subexponentially* γ*-pseudorandom for every constant* $\gamma > 0$*.*

Technical Overview. Let $\mathsf{PRG} = (\mathsf{IdSamp}, \mathsf{Eval})$ 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 assumption over $\mathbb{Z}_p$ to show that the seed is hidden.

Towards this, sPRG "encrypts" the seed $\boldsymbol{\sigma}$ using LPN samples over $\mathbb{Z}_p$ as follows:

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

It follows directly from the LPN over $\mathbb{Z}_p$ assumption that $(\boldsymbol{A}, \boldsymbol{b})$ is pseudorandom and hides $\boldsymbol{\sigma}$. Furthermore, due to the sparsity of LPN noises, the vector $\boldsymbol{\sigma} + \boldsymbol{e}$ differs from $\boldsymbol{\sigma}$

only at a $r = \ell^{-\delta}$ fraction of components – thus it is a sparsely erroneous version of the seed.

Given such "encryption", by applying previous techniques [AJL+19, JLMS19, JLS19, GJLS20] 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 $G^{(1)}$ (that depends on $\boldsymbol{A}$) that has degree d on the public component $\boldsymbol{b}$ and only degree 2 on all possible degree $\lceil \frac{d}{2} \rceil$ monomials in $\boldsymbol{s}$. More precisely,

$$\boldsymbol{y}' = \mathsf{Eval}_I\big(\boldsymbol{\sigma} + \boldsymbol{e}\big) = G^1\big(\boldsymbol{b}\ ,\ (\overline{\boldsymbol{s}}^{\otimes \lceil \frac{d}{2} \rceil})\big), \qquad \overline{\boldsymbol{s}} = \boldsymbol{s}||1 \tag{1}$$

where $\boldsymbol{v}^{\otimes k}$ denotes tensoring the vector $\boldsymbol{v}$ with itself k times, yielding a vector of dimension $\mathsf{dim}(\boldsymbol{v})^k$. In particular, observe that by setting the dimension ℓ of secret $\boldsymbol{s}$ to be sufficiently small, the polynomial $G^{(1)}$ can be expanding; this is done by setting parameters $\ell(n)$ so that $\left(\ell^{\lceil \frac{d}{2} \rceil} + n\right) \ll m(n)$. The reasoning behind comparing the the number of output bits $m = n^\tau$ with the number of field elements in the seed of sPRG is that if $m \gg \mathsf{dim}((\boldsymbol{b}, \overline{\boldsymbol{s}}^{\otimes \lceil \frac{d}{2} \rceil}))$, then, we have polynomial expansion because the the length of the modulus p is at most λ bits which is asymptotically smaller than the parameter n.

However, the new problem is that 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 preprocessed 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. By a simple Markov argument, the number of bad outputs is bounded by $T = mr\log n = \frac{m \log n}{\ell^\delta}$ with probability $1 - o(1)$. Leveraging this sparsity, sPRG corrects bad outputs using the method described below. In the low probability event where there are too many bad outputs (greater than T), it simply outputs 0.

We describe a sequence of ideas that lead to the final correction method, starting with two wrong ideas that illustrate the difficulties we will overcome.

- The first wrong idea is correcting by adding the difference $\mathsf{Corr} = \boldsymbol{y} - \boldsymbol{y}'$ between the correct and erroneous outputs, $\boldsymbol{y} = \mathsf{Eval}_I(\boldsymbol{\sigma})$ and $\boldsymbol{y}' = \mathsf{Eval}_I(\boldsymbol{\sigma} + \boldsymbol{e})$; refer to Corr as the *correction vector*. To obtain the correct output, evaluation can compute the following polynomial $G^{(1)}\big(\boldsymbol{b}\ ,\ (\overline{\boldsymbol{s}}^{\otimes \lceil \frac{d}{2} \rceil})\big) + \mathsf{Corr}$. 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 T elements). More precisely, the j'th output can be computed as $G^{(1)}_j\big(\boldsymbol{b}\ ,\ (\overline{\boldsymbol{s}}^{\otimes \lceil \frac{d}{2} \rceil})\big) + \mathsf{Corr}_j$ if output j is bad and without adding Corr_j otherwise. This fixes expansion, but now the evaluation polynomial depends on the location of bad outputs, which in turn leaks information of the location of LPN noises, and jeopardizes security.

The two wrong ideas illustrate the tension between the expansion and security of 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. Let's first illustrate our ideas in two simple cases.

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[1], it has low rank z. We can then factorize Corr into two matrixes $\mathbf{U}$ and $\mathbf{V}$ of dimensions $\sqrt{m} \times z$ and $z \times \sqrt{m}$ respectively, such that $\mathsf{Corr} = \mathbf{U}\mathbf{V}$, and compute the correct output as follows:

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

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: Evenly spread bad outputs. The above method however cannot handle more than $\sqrt{m}$ bad outputs, whereas the actual number of bad outputs can be up to $T = m(\log n)/\ell^{\delta}$, which can be 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: If we divide the matrix Corr into $\frac{m}{\ell^{\delta}}$ 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 $i \in [\frac{m}{\ell^{\delta}}]$, 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:

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

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. 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 every output bit of PRG depends on the first seed bit. With probability $\frac{1}{\ell^{\delta}}$ it is erroneous and so are all outputs.

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

To overcome this, our final idea is to "force" the even spreading of the bad outputs, by assigning them 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 $B = \frac{mt}{\ell^\delta}$ 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, which is also negligible in λ. 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 the correct output as:

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

$G^{(2)}$ is expanding, because the number of field elements in $\mathbf{U}_i$'s and $\mathbf{V}_i$'s are much smaller than m, namely: $2t^2\ell^{\delta/2}B = O(\frac{m\lambda^3}{\ell^{\delta/2}}) \ll m$. 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 $I' = (I, \boldsymbol{A}, \phi_{\mathsf{bkt}}, \phi_{\mathsf{ind}})$.

Step 3: Zeroize if uneven buckets. Finally, to deal with the low probability event that some bucket is assigned more than $t^2\ell^\delta$ outputs or contains more than t bad outputs, we introduce a new variable called flag. If either of the conditions above occur, our sPRG sets $\mathsf{flag} = 0$ and outputs zero. We then include flag in the public seed and augment the evaluation polynomial as follow:

$$\forall j \in [m],\ \ G^{(3)}_j\Big(\underbrace{(\boldsymbol{b},\ \mathsf{flag})}_{P},\ S\Big) = \mathsf{flag}\cdot G^{(2)}_j\left(\boldsymbol{b},\ S\right)\ .$$

This is our final evaluation polynomial. It has constant degree $d+1$ in the public seed P, degree 2 in the private seed S, and expansion similar to that of $G^{(2)}$. For security, observe that the polynomial $G^{(3)}$ is independent of the location of LPN noises, while the public seed leaks 1-bit of information through flag, which can be simulated efficiently via leakage simulation. Therefore, by the LPN over $\mathbb{Z}_p$ assumption, the seed $\boldsymbol{\sigma}$ of PRG is hidden and the security of PRG ensures that the output is pseudorandom when it is not zeroized. We now proceed to the formal construction and proof.

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

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

- λ is the security parameter,
- n input length to the PRG. n is arbitrary polynomial in λ,
- the stretch τ and degree d of PRG. Set $m = n^{\tau}$,
- the LPN secret dimension $\ell = n^{1/\lceil d/2 \rceil}$, modulus p be a λ bit prime modulus,
- a threshold $T = \frac{m \cdot \log n}{\ell^{\delta}}$ of the number of bad outputs that can be tolerated,
- a slack parameter t used for bounding the capacity of and number of bad outputs in each bucket, set to $t = \lambda$.
- a parameter $B = \frac{m \cdot t}{\ell^{\delta}}$ that indicates the number of buckets used.
- 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 $\boldsymbol{A} \leftarrow \mathbb{Z}_p^{\ell \times n}$. 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 $I' = (I, \boldsymbol{\phi}, \boldsymbol{A})$.

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

- Sample a PRG seed $\boldsymbol{\sigma} \leftarrow \{0, 1\}^{n}$.
- Prepare samples of LPN over $\mathbb{Z}_p$: Sample $\boldsymbol{s} \leftarrow \mathbb{Z}_p^{1 \times \ell}$, $\boldsymbol{e} \leftarrow \mathcal{D}_r^{1 \times n}(p)$, and set

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

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

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

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

- Find indices $j \in [m]$ of outputs that depend on one or more erroneous seed indices. Let Vars_j denote the indices of seed bits that the j'th output of Eval_I depends on. Define:

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

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

- Set $\mathsf{flag} = 0$ if
  1. *Too many* bad *output bits:* $|\mathsf{BAD}| > T$,
  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 $\mathsf{flag} = 1$.

- 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 $\mathsf{flag} = 1$, 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 $\mathsf{flag} = 1$, condition 3 above implies that each $\mathbf{M}_i$ has at most t nonzero entries, and hence rank at most t.
  - If $\mathsf{flag} = 0$, for every $i \in [B]$, set $\mathbf{U}_i$ and $\mathbf{V}_i$ to be 0 matrices.

- Set the public seed to

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

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

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

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

$\boldsymbol{y} \to \mathsf{Eval}'(I', \mathsf{sd})$**:** Compute $\boldsymbol{y} \leftarrow \mathsf{Eval}(I, \boldsymbol{\sigma})$, and output $\boldsymbol{z} = \mathsf{flag} \cdot \boldsymbol{y}$. This computation is done via a polynomial $G^{(3)}_{I'}$ described below that has constant degree $d+1$ in the public seed and only degree 2 in the private seed, that is,

$$\mathsf{Eval}'(I', \mathsf{sd}) = \mathsf{flag} \cdot \boldsymbol{y} = \mathsf{flag} \cdot \mathsf{Eval}_I(\boldsymbol{\sigma}) = G^{(3)}_{I'}(P, S) \ .$$

We next define $G^{(3)}_{I'}$ using intermediate polynomials $G^{(1)}_{I'}$ and $G^{(2)}_{I'}$. For simplicity of notation, we suppress subscript I' below.

- 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

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

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

  $$\begin{aligned} x_i &= \langle \boldsymbol{c}_i, \ \overline{\boldsymbol{s}} \rangle \qquad\qquad && \boldsymbol{c}_i = -\boldsymbol{a}_i^{\mathrm{T}} || b_i, \ \boldsymbol{a}_i \text{ is the } i\text{th column of } \boldsymbol{A} \\ x_v &= \langle \otimes_{i \in v} \boldsymbol{c}_i, \ \otimes_{i \in v} \overline{\boldsymbol{s}} \rangle && \end{aligned}$$

  (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 $G^{(1)}(\boldsymbol{b}, S)$ that computes $\mathsf{Eval}(\boldsymbol{\sigma} + \boldsymbol{e})$:

  $$G^{(1)}_j(\boldsymbol{b}, S) := L_j \left((\langle \otimes_{i \in v} \boldsymbol{c}_i, \ \otimes_{i \in v} \overline{\boldsymbol{s}} \rangle)_{v \in \mathsf{Eval}_j} \right) \ . \qquad (3)$$

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

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

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

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

  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

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

  At last, $G^{(3)}$ has degree $d+1$ in the public seed, and only degree 2 in the private seed, as desired.

Analysis of Stretch. We derive a set of constraints, under which sPRG has polynomial stretch. Recall that PRG output length is $m = n^{\tau}$, degree d, LPN secret dimension $\ell = n^{1/\lceil d/2\rceil}$, modulus $p = O(2^{\lambda})$, 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 $P = (\boldsymbol{b}, \mathsf{flag})$ and has bit length

  $$|P| = O(n \log p) = O(n \cdot \lambda) \ .$$

- The private seed S contains S_1, S_2 as follows:

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

  The bit-lengths are:

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

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

$$|\mathsf{sd}| = |P| + |S_1| + |S_2| = O((n + n^{\tau - \frac{\delta}{2\lceil\frac{d}{2}\rceil}}) \cdot \lambda^4)$$

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

Proof of Pseudorandomness We prove the following proposition which implies that sPRG is γ-pseudorandom for any constant γ.

Proposition 4.1. *Let ℓ, n, r, p be defined as above. For any running time $T = T(\lambda) \in \mathbb{N}$, if*

- $\mathsf{LPN}(\ell, n, r, p)$ *is $(T, \epsilon_{\mathsf{LPN}})$-indistinguishable for advantage $\epsilon_{\mathsf{LPN}} = o(1)$, and*
- PRG *is $(T, \epsilon_{\mathsf{PRG}})$-pseudorandom for advantage $\epsilon_{\mathsf{PRG}} = o(1)$,*

sPRG *satisfies that for every constant $\gamma \in (0, 1)$, the following two distributions are (T, γ)-indistinguishable.*

$$\left\{ (I, \boldsymbol{\phi}, \boldsymbol{A}, \boldsymbol{b}, \mathsf{flag}, \boldsymbol{z}) \ : \ (I,\ \boldsymbol{\phi},\ \boldsymbol{A}) \leftarrow \mathsf{IdSamp}'(1^{n'}),\ (P, S) \leftarrow \mathsf{SdSamp}'(I'),\ \boldsymbol{z} \leftarrow \mathsf{Eval}'(I, \mathsf{sd}) \right\}$$
$$\left\{ (I, \boldsymbol{\phi}, \boldsymbol{A}, \boldsymbol{b}, \mathsf{flag}, \boldsymbol{r}) \ : \ (I,\ \boldsymbol{\phi},\ \boldsymbol{A}) \leftarrow \mathsf{IdSamp}'(1^{n'}),\ (P, S) \leftarrow \mathsf{SdSamp}'(I'),\ \boldsymbol{r} \leftarrow \{0,1\}^m \right\},$$

(Recall that $P = (\boldsymbol{b}, \mathsf{flag})$.)

We start with some intuition behind the proposition. Observe first that if flag is removed, the above two distributions becomes truly indistinguishable. This follows from the facts that i) I and $\boldsymbol{\phi}$ are completely independent of $(\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{z})$ or $(\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{r})$, and ii) $(\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{z})$ and $(\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{r})$ are indistinguishable following from the LPN over $\mathbb{Z}_p$ assumption and the pseudorandomness of PRG. The latter indistinguishability is the heart of the security of sPRG, and is captured in Lemma 4.1 below. Towards the proposition, we need to additional show that publishing flag does not completely destroy the indistinguishability. This follows from the facts that i) flag is only 0 with sub-constant probability, and ii) it can be viewed as a single bit leakage of the randomness used for sampling the rest of the distributions, and can be simulated efficiently by the leakage simulation lemma, Theorem 2.1. The formal proof of the proposition below presents the details.

Lemma 4.1. *Let $G : \{0,1\}^{1\times n} \to \{0,1\}^{1\times m(n)}$ be a $(T, \epsilon_{\mathsf{PRG}})$-secure pseudorandom generator. Assume that $\mathsf{LPN}(\ell, n, r, p)$ is $(T, \epsilon_{\mathsf{LPN}})$-secure. Then the following two distributions are $(T, \epsilon_{\mathsf{LPN}} + \epsilon_{\mathsf{PRG}})$-indistinguishable:*

$$\mathcal{D}_1 = \left\{ (\boldsymbol{A},\ \boldsymbol{b} = \boldsymbol{s} \cdot \boldsymbol{A} + \boldsymbol{e} + \boldsymbol{\sigma},\ G(\boldsymbol{\sigma}))\ :\ \boldsymbol{A} \leftarrow \mathbb{Z}_p^{\ell\times n};\ \boldsymbol{s} \leftarrow \mathbb{Z}_p^{1\times\ell};\ \boldsymbol{e} \leftarrow \mathcal{D}_r^{1\times n}(p);\ \boldsymbol{\sigma} \leftarrow \{0,1\}^{1\times n} \right\}$$
$$\mathcal{D}_2 = \left\{ (\boldsymbol{A},\ \boldsymbol{u},\ \boldsymbol{w})\ :\ \boldsymbol{A} \leftarrow \mathbb{Z}_p^{\ell\times n};\ \boldsymbol{u} \leftarrow \mathbb{Z}_p^{1\times n};\ \boldsymbol{w} \leftarrow \{0,1\}^{1\times m(n)} \right\}$$

Proof. We introduce one intermediate distribution $\mathcal{D}'$ defined as follows:

$$\mathcal{D}' = \left\{ (\boldsymbol{A},\ \boldsymbol{u},\ G(\boldsymbol{\sigma}))\ :\ \boldsymbol{A} \leftarrow \mathbb{Z}_p^{\ell \times n};\ \boldsymbol{u} \leftarrow \mathbb{Z}_p^{1 \times n};\ \boldsymbol{\sigma} \leftarrow \{0,1\}^n \right\}$$

Now observe that $\mathcal{D}'$ is $(T, \epsilon_{\mathsf{LPN}})$-indistinguishable to $\mathcal{D}_1$ following immediately from the $(T, \epsilon_{\mathsf{LPN}})$-indistinguishability of the $\mathsf{LPN}(\ell, n, r, p)$ assumption. Finally, observe that $\mathcal{D}'$ is $(T, \epsilon_{\mathsf{PRG}})$-indistinguishable to $\mathcal{D}_2$ due to $(T, \epsilon_{\mathsf{PRG}})$-security of G. Therefore, the lemma holds. □

Proof of Proposition 4.1. We now list a few hybrids $\mathsf{H}_0, \mathsf{H}_1, \mathsf{H}_2, \mathsf{H}_3$, where the first one corresponds to the first distribution in the proposition, and the last one corresponds to the second distribution in the proposition. We abuse notation to also use H_i to denote the output distribution of the hybrid. Let γ be the claimed advantage of the adversary $\mathcal{A}$, running in time $Tq(\lambda)$ for a polynomial q. Let $\mathcal{D}_{\phi,I}$ denote the the distribution that samples the functions $\boldsymbol{\phi}$.

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

$$\begin{aligned} \text{Sample: } & \boldsymbol{A} \leftarrow \mathbb{Z}_p^{\ell \times n},\ \boldsymbol{s} \leftarrow \mathbb{Z}_p^{1 \times \ell},\ \boldsymbol{e} \leftarrow \mathcal{D}_r^{1 \times n}(p),\ \boldsymbol{\sigma} \leftarrow \{0,1\}^n \\ & I \leftarrow \mathsf{IdSamp}(1^\lambda, 1^n),\ \boldsymbol{y} = \mathsf{Eval}_I(\boldsymbol{\sigma}),\ \boldsymbol{\phi} \leftarrow \mathcal{D}_{\phi,I} \\ \text{Output: } & I,\ \boldsymbol{\phi},\ \boldsymbol{A}, \boldsymbol{b} = \boldsymbol{s}\boldsymbol{A} + \boldsymbol{e} + \boldsymbol{\sigma},\ \mathsf{flag} \cdot \boldsymbol{y} \\ & \text{where } \mathsf{flag} = 1 \text{ iff:} \\ & 1)\ |\mathsf{BAD}| \le T \text{ and,} \\ & 2)\ \forall i \in [B],\ |\phi_{\mathsf{bkt}}^{-1}(i) \cap \mathsf{BAD}| \le t \text{ and,} \\ & 3)\ \forall i \in [B],\ |\phi_{\mathsf{bkt}}^{-1}(i)| \le \ell^\delta \cdot t^2. \end{aligned}$$

Note that the value of flag is correlated with that of $(I, \boldsymbol{\phi}, \boldsymbol{A}, \boldsymbol{b}, \boldsymbol{y})$. Therefore, flag can be viewed as a single-bit leakage of the randomness used for sampling $(I, \boldsymbol{\phi}, \boldsymbol{A}, \boldsymbol{b}, \boldsymbol{y})$.

Hybrid H_1 instead of generating flag honestly, first samples $X = (I, \boldsymbol{\phi}, \boldsymbol{A}, \boldsymbol{b}, \boldsymbol{y})$ honestly, and then invokes the leakage simulation lemma, Lemma 2.1, to simulate flag using X, for $Tq(\lambda) + \mathsf{poly}(\lambda)$ time adversaries with at most $\frac{\gamma}{3}$ advantage. Let Sim be the simulator given by Theorem 2.1.

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

The leakage simulation lemma guarantees that the running time of Sim is bounded by $O((Tq(\lambda) + \mathsf{poly}(\lambda)) \cdot \frac{9}{\gamma^2} \cdot 2^1) = Tq'(\lambda))$ for a fixed polynomial q', and $\mathcal{A}$ cannot distinguish H_0 from H_1 with advantage more than $\frac{\gamma}{3}$.

Claim 4.2. *For any adversary $\mathcal{A}$ running in time $Tq(n)$ for some polynomial q,*

$$|\Pr[\mathcal{A}(\mathsf{H}_0) = 1] - \Pr[\mathcal{A}(\mathsf{H}_1) = 1]| \le \frac{\gamma}{3}\,.$$

Furthermore, the running time of Sim *is $Tq'(\lambda)$ for some polynomial q'.*

This claim is immediate from Lemma 2.1.

Hybrid H_2 samples $\boldsymbol{A}, \boldsymbol{b}$ and $\boldsymbol{y}$ uniformly and randomly.

$$\begin{aligned}\text{Sample: } & \underline{\boldsymbol{A} \leftarrow \mathbb{Z}_p^{\ell \times n},\ \boldsymbol{b} \leftarrow \mathbb{Z}_p^{1 \times n}} \\ & I \leftarrow \mathsf{IdSamp}(1^\lambda, 1^n),\ \underline{\boldsymbol{y} \leftarrow \{0,1\}^m},\ \boldsymbol{\phi} \leftarrow \mathcal{D}_{\phi,I} \\ \text{Output: } & I,\ \boldsymbol{\phi},\ \boldsymbol{A},\ \boldsymbol{b},\ \mathsf{flag} \cdot \boldsymbol{y} \\ & \text{where } \mathsf{flag} = \mathsf{Sim}(I, \boldsymbol{\phi}, \boldsymbol{A}, \boldsymbol{b}, \boldsymbol{y})\end{aligned}$$

Lemma 4.1 shows that $(\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{y})$ generated honestly as in H_1 and $(\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{y})$ sampled all at random as in H_2 are indistinguishable, due to the LPN assumption and the pseudorandomness of PRG. Here the adversary $\mathcal{A}$ runs in time $Tq(\lambda)$ and the simulator Sim runs in time $Tq'(\lambda)$ time, for polynomials q, q'. Thus, we get

Claim 4.3. *For any adversary $\mathcal{A}$, running in time T, if $\mathsf{LPN}(\ell, n, r, p)$ is $(T, \epsilon_{\mathsf{LPN}})$-secure and PRG satisfies $(T, \epsilon_{\mathsf{PRG}})$-pseudorandomness, then,*

$$|\Pr[\mathcal{A}(\mathsf{H}_1) = 1] - \Pr[\mathcal{A}(\mathsf{H}_2) = 1]| \leq \epsilon_{\mathsf{PRG}} + \epsilon_{\mathsf{LPN}}$$

This claim follows immediately from Lemma 4.1.

Hybrid H_3 no longer generates flag and simply outputs the random string $\boldsymbol{y}$ instead of $\mathsf{flag} \cdot \boldsymbol{y}$.

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

Observe that H_2 and H_3 are only distinguishable when $\mathsf{flag} = 0$ in H_2. By bounding the probability of $\mathsf{flag} = 0$ in H_2, we can show that

Claim 4.4. *For any adversary $\mathcal{A}$,*

$$|\Pr[\mathcal{A}(\mathsf{H}_2) = 1] - \Pr[\mathcal{A}(\mathsf{H}_3) = 1]| \leq \frac{\gamma}{2}$$

The formal proof of this lemma is provided below.

Combining the hybrids above, we conclude that $\mathcal{A}$ cannot distinguish H_0 and H_3 with advantage more than $\frac{5 \cdot \gamma}{6} + \epsilon_{\mathsf{PRG}} + \epsilon_{\mathsf{LPN}} < \gamma$, which gives a contradiction. Therefore, the indistinguishability stated in the proposition holds. We now complete the final remaining piece – the proof of Claim 4.4.

Proof of Claim 4.4. This indistinguishability is statistical. We start with showing that the probability that $\mathsf{flag} = 0$ in H_0 is $O(\frac{1}{\log n})$. Towards this, we bound probability of all three conditions for setting $\mathsf{flag} = 0$ and then apply the union bound.

- $\Pr[|\mathsf{BAD}| > T] \le O(\frac{1}{\log n})$. Observe that by the fact that Eval_I has constant locality in $\boldsymbol{\sigma}$, the probability that any single output bit $j \in [m]$ is bad is bounded by $O(r) = \frac{O(1)}{\ell^\delta}$, where r is the rate of LPN noises. Therefore, the expected number of bad output bits is

  $$\mathbb{E}[|\mathsf{BAD}|] = \frac{O(1)m}{\ell^\delta}$$

  Thus by Markov's inequality,

  $$\mathsf{Pr}[|\mathsf{BAD}| > T] \le \frac{1}{T} \cdot \frac{O(1)m}{\ell^\delta \cdot T} = \frac{O(1)}{\log n} \ .$$

  The last equality follows from the fact that $T = \frac{m \cdot \log n}{\ell^\delta}$.

- For any $i \in [B]$, $\Pr_{\phi_{\mathsf{bkt}}}\left[|\phi_{\mathsf{bkt}}^{-1}(i) \cap \mathsf{BAD}| > t \mid |\mathsf{BAD}| \le T\right] \le \mathsf{negl}(n)$. Suppose $|\mathsf{BAD}| = T'$ where $T' \le T$, and since $\phi_{\mathsf{bkt}} : [m] \to [B]$ is a random function, we have:

  $$\begin{aligned} \Pr_{\phi_{\mathsf{bkt}}}\left[|\phi_{\mathsf{bkt}}^{-1}(i) \cap \mathsf{BAD}| > t \mid |\mathsf{BAD}| = T'\right] &\le \binom{T'}{t} \cdot \frac{1}{B^t} \\ &\le \left(\frac{e \cdot T'}{t}\right)^t \cdot \frac{1}{B^t} && \text{by Stirling's approximation} \\ &\le \left(\frac{e}{t}\right)^t \le e^{-t} && \text{by } T' < T < B \\ &= \mathsf{negl}(\lambda) && \text{by } t = \lambda \end{aligned}$$

- For any $i \in B$, $\Pr_{\phi_{\mathsf{bkt}}}[|\phi_{\mathsf{bkt}}^{-1}(i)| > \ell^\delta \cdot t^2] \le \mathsf{negl}(\lambda)$. Since ϕ_{bkt} is a random function,

  $$\begin{aligned} \Pr_{\phi_{\mathsf{bkt}}}\left[|\phi_{\mathsf{bkt}}^{-1}(i)| > t^2 \cdot \ell^\delta\right] &\le \binom{m}{\ell^\delta \cdot t^2} \cdot \left(\frac{1}{B}\right)^{\ell^\delta \cdot t^2} \\ &\le \left(\frac{e \cdot m}{\ell^\delta \cdot t^2}\right)^{\ell^\delta \cdot t^2} \cdot \left(\frac{1}{B}\right)^{\ell^\delta \cdot t^2} && \text{by Stirling's approximation} \\ &= \left(\frac{e \cdot m}{B \cdot \ell^\delta \cdot t^2}\right)^{\ell^\delta \cdot t^2} \le \left(\frac{1}{t^2}\right)^{\ell^\delta \cdot t^2} && \text{by } B = \frac{mt}{\ell^\delta} > \frac{em}{\ell^\delta} \\ &\le t^{-2t^2} = \mathsf{negl}(\lambda) && \text{by } \ell^\delta > 1 \text{ and } t = \lambda \end{aligned}$$

Applying the three observations above, from a union bound it follows that $\Pr[\mathsf{flag} = 0] = O(\frac{1}{\log n})$.

Next, for adversaries of run time $Tq(\lambda)$, Claim 4.2 shows that H_0 and H_1 cannot be distinguished with advantage more than $\frac{\gamma}{3}$, and Claim 4.3 shows that H_1 and H_2 cannot be distinguished with advantage more than $\epsilon_{\mathsf{PRG}} + \epsilon_{\mathsf{LPN}}$, which is sub-constant. Therefore, the probability that $\mathsf{flag} = 0$ in H_2 is upper bounded by

$$\mathsf{Pr}[\mathsf{flag} = 0 \text{ in } \mathsf{H}_2] \le \frac{O(1)}{\log n} + \frac{\gamma}{3} + \epsilon_{\mathsf{PRG}} + \epsilon_{\mathsf{LPN}} \le \frac{\gamma}{2} \ .$$

Finally, we upper bound the statistical distance between H_2 and H_3, which is

$$SD(\mathsf{H}_2,\mathsf{H}_3)=\frac{1}{2}\cdot\sum_{(I,\boldsymbol{\phi},\boldsymbol{A},\boldsymbol{b},\boldsymbol{y})}\Big|\Pr[\mathsf{H}_2=(I,\boldsymbol{\phi},\boldsymbol{A},\boldsymbol{b},\boldsymbol{y})]-\Pr[\mathsf{H}_3=(I,\boldsymbol{\phi},\boldsymbol{A},\boldsymbol{b},\boldsymbol{y})]\Big|\ .$$

For $b\in\{0,1\}$, let F_b be the set of tuples $(I,\boldsymbol{A},\boldsymbol{b},\boldsymbol{y})$ that generate $\mathsf{flag}=b$ through Sim,

$$F_b=\{(I,\boldsymbol{\phi},\boldsymbol{A},\boldsymbol{b},\boldsymbol{y})\mid\mathsf{Sim}((I,\boldsymbol{\phi},\boldsymbol{A},\boldsymbol{b},\boldsymbol{y})=b\}\ .$$

Then, we have:

$$\begin{aligned}SD(\mathsf{H}_2,\mathsf{H}_3)&=\frac{1}{2}\cdot\sum_{(I,\boldsymbol{\phi},\boldsymbol{A},\boldsymbol{b},\boldsymbol{y})\in F_0}\Big|\Pr[\mathsf{H}_2=(I,\boldsymbol{\phi},\boldsymbol{A},\boldsymbol{b},\boldsymbol{y})]-\Pr[\mathsf{H}_3=(I,\boldsymbol{\phi},\boldsymbol{A},\boldsymbol{b},\boldsymbol{y})]\Big|\\&+\frac{1}{2}\cdot\sum_{(I,\boldsymbol{\phi},\boldsymbol{A},\boldsymbol{b},\boldsymbol{y})\in F_1}\Big|\Pr[\mathsf{H}_2=(I,\boldsymbol{\phi},\boldsymbol{A},\boldsymbol{b},\boldsymbol{y})]-\Pr[\mathsf{H}_3=(I,\boldsymbol{\phi},\boldsymbol{A},\boldsymbol{b},\boldsymbol{y})]\Big|\\&=\frac{1}{2}\cdot\sum_{(I,\boldsymbol{\phi},\boldsymbol{A},\boldsymbol{b},\boldsymbol{y})\in F_0}\Big|\Pr[\mathsf{H}_2=(I,\boldsymbol{\phi},\boldsymbol{A},\boldsymbol{b},\boldsymbol{y})]-\Pr[\mathsf{H}_3=(I,\boldsymbol{\phi},\boldsymbol{A},\boldsymbol{b},\boldsymbol{y})]\Big|\\&\leq\Pr[\mathsf{flag}=0\text{ in }\mathsf{H}_2]\leq\frac{\gamma}{2}\end{aligned}$$

where the second equality follows from the fact that in H_2 and H_3 the probability of outputing a tuple $(I',\boldsymbol{\phi},\boldsymbol{A},\boldsymbol{b},\boldsymbol{y})$ that belongs to F_1, or equivalently generates $\mathsf{flag}=1$ via Sim, is the same. This concludes the claim.

□

□

5 Bootstrapping to Indistinguishability Obfuscation

We now describe a pathway to $i\mathcal{O}$ and FE for all circuits.

From Structured-Seed PRG to Perturbation Resilient Generator. Starting from structured-seed PRG, we show how to construct perturbation resilient generators, denoted as ΔRG. ΔRG is the key ingredient in several recent $i\mathcal{O}$ constructions [AJL+19, JLMS19, JLS19]. Roughly speaking, they have the same syntax as structured-seed PRGs with the notable difference that it has integer outputs $\boldsymbol{y}$ of polynomial magnitude; further, they only satisfy weak pseudorandomness called *perturbation resilience* guaranteeing that $\boldsymbol{y}+\boldsymbol{\beta}$ for arbitrary adversarially chosen small integer vector $\boldsymbol{\beta}$ is weakly indistinguishable from $\boldsymbol{y}$ itself. The formal definition of ΔRG is provided in Definition 5.1 in Section 5.1.

Theorem 5.1 (sPRG to ΔRG, proven in Section 5.1)**.** *Let $\lambda\in\mathbb{N}$ be the security parameter, $\gamma\in(0,1)$, and $\tau>1$. Assume the existence of a (subexponentially) γ-pseudorandom* sPRG *in $(\mathbb{C},\deg\ d)$ with stretch τ. For any constant $0<\tau'<\tau$, there exists a (subexponentially) $(2\gamma+O(\frac{1}{\lambda}))$-perturbation resilient* ΔRG *in $(\mathbb{C},\deg\ d)$ with a stretch τ'.*

From Perturbation Resilient Generator to Weak FE for NC^0. It was shown in [AJL$^+$19, JLMS19, JLS19] that $\Delta\mathsf{RG}$, along with SXDH, LWE and PRG in NC^0, can be used to construct a secret-key functional encryption scheme for NC^0 circuits. The FE scheme supports only a *single secret key* for a function with multiple output bits, has *weak* indistinguishability security, and has ciphertexts whose sizes grow sublinearly in the circuit size and linearly in the input length. Formal definitions of functional encryption schemes are provided in B.

Theorem 5.2 ([AJL$^+$19, JLMS19, JLS19]). *Let $\gamma \in (0,1)$, $\epsilon > 0$, and $D \in \mathbb{N}$ be arbitrary constants. Let λ be a security parameter, p be an efficiently samplable λ bit prime, and $k = k(\lambda)$ be a large enough positive polynomial in λ. Assume (subexponential) hardness of*

- *the* SXDH *assumption with respect to a bilinear groups of order p,*
- *the* LWE *assumption with modulus-to-noise ratio 2^{k^ϵ} where $k = k(\lambda)$ is the dimension of the secret,*
- *the existence of γ-secure perturbation resilient generators $\Delta\mathsf{RG} \in (\mathsf{arith}\text{-}\mathsf{NC}^0, \deg\ 2)$ over $\mathbb{Z}_p$ with polynomial stretch.*

There exists a secret-key functional encryption scheme for NC^0 circuits with multilinear degree D over $\mathbb{Z}$, having

- *1-key, weakly selective, (subexponential) $(\gamma + \mathsf{negl})$-indistinguishability-security, and*
- sublinearly compact ciphertext with linear dependency on input length, *that is, ciphertext size is $|\mathsf{ct}| = \mathsf{poly}(\lambda)(l + S^{1-\sigma})$, where l is the input length, S the maximum size of the circuits supported, σ is some constant in $(0,1)$, and* poly *depends on D.*

For convenient reference, the construction is recalled in Section B.

From weak FE for NC^0 to Full-Fledged FE for All Polynomial Size Circuits Starting from the above weak version of secret key functional encryption scheme – weak function class NC^0, weak security, and weak compactness – we apply known transformations to obtain a full-fledged *public key* FE scheme for polynomial size circuits, satisfying adaptive collusion resistant security, and having full compactness.

Theorem 5.3 (Strengthening FE). *Let $\gamma \in (0,1)$. Let $\lambda \in \mathbb{N}$ be a security parameter and $k(\lambda)$ be a large enough positive polynomial. Assume the (subexponential) hardness of*

- *the* LWE *assumption with modulus-to-noise ratio 2^{k^ϵ} where $k = k(\lambda)$ is the dimension of the secret, and*
- *the existence of Boolean* PRGs *in NC^0 with polynomial stretch and multilinear degree $d \in \mathbb{N}$ over $\mathbb{Z}$.*

There are the following transformations:

1. STARTING POINT.

   Suppose there is a secret-key functional encryption scheme for NC^0 *circuits with multilinear degree* $(3d+2)$ *over* $\mathbb{Z}$*, having 1-key, weakly selective, (subexponential)* γ*-indistinguishability security, and sublinearly compact ciphertext and linear dependency on input length.*

2. LIFTING FUNCTION CLASS [AJS15, LV16, LIN16].

   There exists a secret-key functional encryption scheme for polynomial size circuits, having 1-key, weakly selective, (subexponential) $(\gamma + \mathsf{negl})$*-indistinguishability security, and sublinearly compact ciphertexts, that is,* $|\mathsf{ct}| = \mathsf{poly}(\lambda, l)S^{1-\sigma}$*.*

3. SECURITY AMPLIFICATION [AJS18, AJL+19, JKMS20].

   There exists a secret-key functional encryption scheme for polynomial-size circuits, having 1-key, weakly selective, (subexponentially) $(\mathsf{negl}$*-)indistinguishability security, and sublinearly compact ciphertexts.*

4. SECRET KEY TO PUBLIC KEY, AND SUBLINEAR CIPHERTEXT TO SUBLINEAR ENCRYPTION TIME [BNPW16, LPST16, GKP+13].

   There exists a public-key functional encryption scheme for polynomial size circuits, having 1-key, weakly selective, (subexponentially) indistinguishability security, and sublinear encryption time, $T_{\mathsf{Enc}} = \mathsf{poly}(\lambda, l)S^{1-\sigma}$*.*

5. 1-KEY TO COLLUSION RESISTANCE [GS16, LM16, KNT18]

   There exists a public-key functional encryption scheme for polynomial-size circuits, having collusion resistant, adaptive, (subexponentially) indistinguishability security, and encryption time $\mathsf{poly}(\lambda, l)$*.*

FE to IO Transformation Finally, we rely on the FE to IO transformation to obtain $i\mathcal{O}$.

Theorem 5.4 ([AJ15, BV15a])**.** *Assume the existence of a public-key functional encryption scheme for polynomial-size circuits, having 1-key, weakly selective, subexponentially indistinguishability security, and sublinear encryption time. Then, (subexponentially secure)* $i\mathcal{O}$ *for polynomial size circuits exists.*

Putting Pieces Together Combining Theorem 4.1, Theorem 5.1, Theorem 5.2, Theorem 5.3, and Theorem 5.4, we get our main result:

Theorem 5.5. *Let* $\tau > 1$*,* $\epsilon, \delta \in (0,1)$*, and* $d \in \mathbb{N}$ *be arbitrary constants. Let* $\lambda \in \mathbb{N}$ *be a security parameter,* p *be an efficiently samplable* λ *bit prime, and* $n = n(\lambda)$ *and* $k = k(\lambda)$ *be large enough positive polynomials in the security parameter. Assume sub-exponential hardness of the following assumptions:*

- *the* LWE *assumption with modulus-to-noise ratio* 2^{k^ϵ} *where* k *is the dimension of the secret,*
- *the* SXDH *assumption with respect to bilinear groups of prime order* p*,*

- *the existence of a Boolean* PRG *in* NC^0 *with polynomial stretch and multilinear degree* d *over* $\mathbb{Z}$*, and*
- *the* $\mathsf{LPN}(\ell, n, \ell^{-\delta}, p)$ *where* $\ell = n^{\frac{1}{\lceil \frac{d}{2} \rceil}}$.

Then, (subexponentially secure) indistinguishability obfuscation for all polynomial-size circuits exists. Further, assuming only polynomial security of these assumptions, there exists collusion resistant, adaptive, and compact public-key functional encryption for all circuits.

5.1 Perturbation Resilient Generators

We recall the definition of perturbation resilient generators from [AJL+19, JLMS19, JLS19].

Definition 5.1 (Syntax of Perturbation Resilient Generators ($\Delta\mathsf{RG}$) [AJL+19, JLMS19, JLS19])**.** *Let* τ *be a positive constant. A perturbation resilient generator* $\Delta\mathsf{RG}$ *with stretch* τ *is defined by the following PPT algorithms:*

- $\mathsf{SetupPoly}(1^\lambda, 1^n, 1^B)$: *takes as input the security parameter* λ*, a seed length parameter* n*, and a bound* B*, samples a function index* I*.*
- $\mathsf{SetupSeed}(I)$: *samples two binary strings, a public seed and a private seed,* $\mathsf{sd} = (P, S)$*. The combined length of these strings is* $n \cdot \mathsf{poly}(\lambda, \log B)$*.*
- $\mathsf{Eval}(I, \mathsf{sd})$: *takes as input the index* I *and the seed* sd *and computes a string in* $\mathbb{Z}^m \cap [-\,\mathsf{poly}(n, B, \lambda),\ \mathsf{poly}(n, B, \lambda)]^m$ *for some fixed polynomial* poly*.*

Remark 5.1. Similar to an sPRG, we say that $\Delta\mathsf{RG}$ has polynomial stretch if above $\tau > 1$ for some constant τ.

Remark 5.2. Note that in the definition proposed by [JLMS19, JLS19], the $\mathsf{SetupSeed}$ algorithm was not given as input I, however, their results still hold even if $\mathsf{SetupSeed}$ is given I as input.

Definition 5.2 (Security of $\Delta\mathsf{RG}$ [AJL+19, JLMS19, JLS19])**.** *A perturbation resilient generator* $\Delta\mathsf{RG}$ *satisfies*

(T, γ)**-perturbation resilience:** *For every* $n = n(\lambda)$ *a positive non-constant polynomial in the security parameter* λ*, and* $B = B(\lambda, n)$ *a positive non-constant polynomial in* λ *and* n*, and every sequence* $\{\boldsymbol{\beta} = \boldsymbol{\beta}_\lambda\}$*, where* $\boldsymbol{\beta} \in \mathbb{Z}^m \cap [-B, B]^m$*, we require that the following two distributions are* $(T(\lambda), \gamma(\lambda))$*-indistinguishable:*

$$\{(I,\ P,\ \mathsf{Eval}(I, \mathsf{sd}, B)) \mid I \leftarrow \mathsf{SetupPoly}(1^\lambda, 1^n, 1^B),\ \mathsf{sd} = (S, P) \leftarrow \mathsf{SetupSeed}(I)\}$$
$$\{(I,\ P,\ \mathsf{Eval}(I, \mathsf{sd}, B) + \boldsymbol{\beta}) \mid I \leftarrow \mathsf{SetupPoly}(1^\lambda, 1^n, 1^B),\ \mathsf{sd} = (S, P) \leftarrow \mathsf{SetupSeed}(I)\}$$

Definition 5.3 (Complexity and degree of $\Delta\mathsf{RG}$)**.** *Let* $d \in \mathbb{N}$*, let* $\lambda \in \mathbb{N}$ *and* $n = n(\lambda)$ *be arbitrary positive non-constant 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* $\Delta\mathsf{RG}$ *has complexity* $\mathbb{C}$ *in the public seed and degree* d *in private seed over* $\mathbb{Z}_p$*, denoted as,* $\Delta\mathsf{RG} \in (\mathbb{C},\ \mathit{deg}\ d)$*, if for any* *polynomial $B(n, \lambda)$ and every I in the support of* $\mathsf{SetupPoly}(1^\lambda, 1^n, 1^B)$*, 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* $\mathsf{SetupSeed}(I)$*, 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 now prove the following proposition, which immediately implies Theorem 5.1.

Proposition 5.1. *Assume the existence of a (T, γ)-pseudorandom structured seed PRG,* sPRG*, in $(\mathbb{C}, \deg\ d)$ with a stretch of $\tau > 0$. Then for any constant $0 < \tau' < \tau$, there exists a $(T, 2 \cdot \gamma + O(\frac{1}{\lambda}))$-perturbation resilient generator,* $\Delta\mathsf{RG}$ *in $(\mathbb{C}, \deg\ d)$ with a stretch τ'.*

Proof. Let sPRG be the given structured-seed PRG with stretch τ. The construction of $\Delta\mathsf{RG}$ is as follows.

- $\Delta\mathsf{RG}.\mathsf{SetupPoly}(1^\lambda, 1^n, 1^B)$: Run $\mathsf{sPRG}.\mathsf{IdSamp}(1^\lambda, 1^n) \to I'$, and output $I = (I', B, \lambda, n)$.
- $\Delta\mathsf{RG}.\mathsf{SetupSeed}(I)$: Run $\mathsf{sPRG}.\mathsf{SdSamp}(I') \to (P, S)$ and output $\mathsf{sd} = (P, S)$.
- $\Delta\mathsf{RG}.\mathsf{Eval}(I, \mathsf{sd})$: Compute $\boldsymbol{z} \leftarrow \mathsf{sPRG}.\mathsf{Eval}(I', \mathsf{sd})$ where $\boldsymbol{z} \in \{0,1\}^{n^\tau}$. Let $m' = n^{\tau'}$ and $t = \lceil \log_2(\lambda \cdot n^{\tau'} \cdot B) \rceil$.
  - If $m < m't$, there are not enough bits in the output of sPRG. Set $\boldsymbol{y} = \mathbf{0}^{1 \times m'}$
  - Otherwise, for every $i \in [m']$, set $y_i = \sum_{j \in [t]} 2^{j-1} \cdot z_{(i-1) \cdot t + j}$.

  Output $\boldsymbol{y}$.

Stretch: The output length is exactly $m' = n^{\tau'}$, while the seed length is identical to that of sPRG, namely $n\,\mathsf{poly}(\lambda)$, as desired.

Further, observe that the output of $\Delta\mathsf{RG}$ is set to 0 when there are not enough bits in the output of sPRG, namely $m < m't$. It is easy to see that for arbitrary non-constant positive polynomials $n = n(\lambda)$ and $B = B(\lambda, n)$, it holds that $t = O(\log \lambda)$ and hence for any $0 < \tau' < \tau$, $m = n^\tau \geq m't = n^{\tau'} t$ for sufficiently large λ. In this case, the output of $\Delta\mathsf{RG}$ is formed by the output of sPRG.

Complexity: We note that $\Delta\mathsf{RG}$ is in $(\mathbb{C}, \deg\ d)$. In the case that $m \geq m't$, $\Delta\mathsf{RG}.\mathsf{Eval}(I, \mathsf{sd})$ outputs $\boldsymbol{y}$ where $y_i = \sum_{j \in [t]} 2^{j-1} \cdot z_{(i-1) \cdot t + j}$, and $\boldsymbol{z} = \mathsf{sPRG}.\mathsf{Eval}(I', \mathsf{sd})$. Since each y_i is a linear function of $\boldsymbol{z}$ and each z_i is degree d in S, $\boldsymbol{y}$ is also degree d in S. Further since each z_i is linear in $\overline{P} = \mathsf{Process}_I(P)$ and $\mathsf{Process}_I \in \mathbb{C}$, $\boldsymbol{y}$ is also linear in $\overline{P} = \mathsf{Process}_I(P)$. In the other case that $m < m't$, the output $\boldsymbol{y} = \mathbf{0}^{1 \times m'}$ and had degree 0 in both P and S. Overall, $\Delta\mathsf{RG} \in (\mathbb{C}, \deg\ d)$.

$(T, 2\cdot\gamma + O(\frac{1}{\lambda}))$**-perturbation resilience:** Fix a sufficiently large $\lambda \in \mathbb{N}$, positive non-constant polynomials $n = n(\lambda)$, $B(\lambda, n)$and $\beta = \beta_\lambda \in \mathbb{Z}^m \cap [-B, B]^m$, and $t = \log_2(\lambda \cdot n^{\tau'} \cdot B)$. We now show the perturbation resilience of ΔRG through a sequence of hybrids.

Hybrid H_0**:** In this hybrid, we give to the adversary,

$$\forall i \in [m'],\ y_i = \sum_{j\in[t]} 2^{j-1} \cdot z_{(i-1)\cdot t+j} + \beta_i\ , \qquad \boldsymbol{z} = \mathsf{sPRG.Eval}(I', \mathsf{sd})\ ,$$

along with the public index I and the public part of the seed P. As observed above, when n and B are positive non-constant polynomials, and λ is sufficiently large, it always holds that $m \geq m't$ and the output of ΔRG is non-zero and formed as above. Thus, this hybrid corresponds to the first challenge distribution in the security definition of ΔRG (Definition 5.2).

Hybrid H_1**:** In this hybrid, we change $\boldsymbol{y}$ to

$$y_i = \sum_{j\in[t]} 2^{j-1} \cdot r_{(i-1)\cdot t+j} + \beta_i\ \ , \qquad \boldsymbol{r} \leftarrow \{0,1\}^{n^\tau}\ .$$

This hybrid is (T,γ)-indistinguishable to hybrid H_0 by the (T,γ)-pseudorandomness of sPRG.

Hybrid H_2**:** In this hybrid, we change $\boldsymbol{y}$ to

$$y_i = u_i + \beta_i\ , \qquad u_i \leftarrow [0, 2^t - 1]\ .$$

This hybrid is identical to hybrid H_1.

Hybrid H_3**:** In this hybrid, we change $\boldsymbol{y}$ to

$$y_i = u_i\ , \qquad u_i \leftarrow [0, 2^t - 1]\ .$$

This hybrid is statistically close to hybrid H_2 with the statistical distance bounded by $O(m' \cdot \frac{B}{2^t-1}) = O(\frac{1}{n})$. This is because each u_i is uniform between $[0, 2^t - 1]$ and $|\beta_i| \leq B$.

Hybrid H_4**:** In this hybrid, we change $\boldsymbol{y}$ to

$$y_i = \sum_{j\in[t]} 2^{j-1} \cdot r_{(i-1)\cdot t+j}\ , \qquad \boldsymbol{r} \leftarrow \{0,1\}^{n^\tau}\ .$$

The hybrid above is identical to hybrid H_3.

Hybrid H_5**:** In this hybrid, we give to the adversary,

$$y_i = \sum_{j\in[t]} 2^{j-1} \cdot z_{(i-1)\cdot t+j}\ , \qquad \boldsymbol{z} = \mathsf{sPRG.Eval}(I', \mathsf{sd})\ .$$

This hybrid is (T,γ)-indistinguishable to hybrid H_4 by the (T,γ)-pseudorandomness of sPRG. By the same argument as in hybrid H_0, we have $m \geq m't$ and the output of ΔRG is non-zero and exactly as above. Thus, this corresponds to the second challenge distribution in Definition 5.2.

By a hybrid argument, we get that the total advantage in distinguishing the two challenge distributions in the security definition of $\Delta\mathsf{RG}$ is bounded by $2 \cdot \gamma + O(\frac{1}{\lambda})$. This concludes the proof. $\square$

6 Acknowledgements

We would like to thank Stefano Tessaro and James Bartusek for helpful discussions. 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), 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, the National Science Foundation, NTT Research, Google, or the U.S. Government.

7 References

[AAB15] Benny Applebaum, Jonathan Avron, and Christina Brzuska. Arithmetic cryptography: Extended abstract. In Tim Roughgarden, editor, *ITCS 2015*, pages 143–151. ACM, January 2015.

[ABR12] Benny Applebaum, Andrej Bogdanov, and Alon Rosen. A dichotomy for local small-bias generators. In Ronald Cramer, editor, *TCC 2012*, volume 7194 of *LNCS*, pages 600–617. Springer, Heidelberg, March 2012.

[ADI^{+}17] Benny Applebaum, Ivan Damgård, Yuval Ishai, Michael Nielsen, and Lior Zichron. Secure arithmetic computation with constant computational overhead. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part I*, volume 10401 of *LNCS*, pages 223–254. Springer, Heidelberg, August 2017.

[AGIS14] Prabhanjan Ananth, Divya Gupta, Yuval Ishai, and Amit Sahai. Optimizing obfuscation: Avoiding Barrington's theorem. In *ACM CCS*, pages 646–658, 2014.

[Agr19] Shweta Agrawal. Indistinguishability obfuscation without multilinear maps: New methods for bootstrapping and instantiation. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part I*, volume 11476 of *LNCS*, pages 191–225. Springer, Heidelberg, May 2019.

[AJ15] Prabhanjan Ananth and Abhishek Jain. Indistinguishability obfuscation from compact functional encryption. In *Advances in Cryptology–CRYPTO 2015*, pages 308–326. Springer, 2015.

[AJL^{+}19] Prabhanjan Ananth, Aayush Jain, Huijia Lin, Christian Matt, and Amit Sahai. Indistinguishability obfuscation without multilinear maps: New paradigms via low degree weak pseudorandomness and security amplification. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part III*, volume 11694 of *LNCS*, pages 284–332. Springer, Heidelberg, August 2019.

[AJS15] Prabhanjan Ananth, Abhishek Jain, and Amit Sahai. Indistinguishability obfuscation from functional encryption for simple functions. *Eprint*, 730:2015, 2015.

[AJS18] Prabhanjan Ananth, Aayush Jain, and Amit Sahai. Indistinguishability obfuscation without multilinear maps: io from lwe, bilinear maps, and weak pseudorandomness. *IACR Cryptology ePrint Archive*, 2018:615, 2018.

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

[Ale03] Michael Alekhnovich. More on average case vs approximation complexity. In *44th FOCS*, pages 298–307. IEEE Computer Society Press, October 2003.

[AP20] Shweta Agrawal and Alice Pellet-Mary. Indistinguishability obfuscation without maps: Attacks and fixes for noisy linear FE. In Vincent Rijmen and Yuval Ishai, editors, *EUROCRYPT 2020, Part I*, LNCS, pages 110–140. Springer, Heidelberg, May 2020.

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

[AS17] Prabhanjan Ananth and Amit Sahai. Projective arithmetic functional encryption and indistinguishability obfuscation from degree-5 multilinear maps. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part I*, volume 10210 of *LNCS*, pages 152–181. Springer, Heidelberg, April / May 2017.

[BBKK17] Boaz Barak, Zvika Brakerski, Ilan Komargodski, and Pravesh Kothari. Limits on low-degree pseudorandom generators (or: Sum-of-squares meets program obfuscation). *Electronic Colloquium on Computational Complexity (ECCC)*, 24:60, 2017.

[BCG$^+$19] Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, Peter Rindal, and Peter Scholl. Efficient two-round OT extension and silent non-interactive secure computation. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019*, pages 291–308. ACM Press, November 2019.

[BCGI18] Elette Boyle, Geoffroy Couteau, Niv Gilboa, and Yuval Ishai. Compressing vector OLE. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *ACM CCS 2018*, pages 896–912. ACM Press, October 2018.

[BDGM20] Zvika Brakerski, Nico Dottling, Sanjam Garg, and Guilio Malavolta. Candidate io from homomorphic encryption schemes. In *EUROCRYPT*, 2020.

[BFM14] Christina Brzuska, Pooya Farshim, and Arno Mittelbach. Indistinguishability obfuscation and UCEs: The case of computationally unpredictable sources. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 188–205. Springer, Heidelberg, August 2014.

[BGdMM05] Lucas Ballard, Matthew Green, Breno de Medeiros, and Fabian Monrose. Correlation-resistant storage via keyword-searchable encryption. *IACR Cryptol. ePrint Arch.*, 2005:417, 2005.

[BGG$^+$14] Dan Boneh, Craig Gentry, Sergey Gorbunov, Shai Halevi, Valeria Nikolaenko, Gil Segev, Vinod Vaikuntanathan, and Dhinakaran Vinayagamurthy. Fully key-homomorphic encryption, arithmetic circuit ABE and compact garbled circuits. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology - EUROCRYPT 2014 - 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings*, volume 8441 of *Lecture Notes in Computer Science*, pages 533–556. Springer, 2014.

[BGG$^+$18] Dan Boneh, Rosario Gennaro, Steven Goldfeder, Aayush Jain, Sam Kim, Peter M. R. Rasmussen, and Amit Sahai. Threshold cryptosystems from threshold fully homomorphic encryption. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part I*, volume 10991 of *Lecture Notes in Computer Science*, pages 565–596. Springer, 2018.

[BGH$^+$15] Zvika Brakerski, Craig Gentry, Shai Halevi, Tancrede Lepoint, Amit Sahai, and Mehdi Tibouchi. Cryptanalysis of the quadratic zero-testing of GGH. Cryptology ePrint Archive, Report 2015/845, 2015. `http://eprint.iacr.org/`.

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

[BGI$^+$01b] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In *Advances in Cryptology - CRYPTO 2001, 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19-23, 2001, Proceedings*, pages 1–18, 2001.

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

[BGV12] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. In Shafi Goldwasser, editor, *ITCS 2012*, pages 309–325. ACM, January 2012.

[BHJ$^+$19] Boaz Barak, Samuel B. Hopkins, Aayush Jain, Pravesh Kothari, and Amit Sahai. Sum-of-squares meets program obfuscation, revisited. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part I*, volume 11476 of *LNCS*, pages 226–250. Springer, Heidelberg, May 2019.

[BIJ$^+$20] James Bartusek, Yuval Ishai, Aayush Jain, Fermi Ma, Amit Sahai, and Mark Zhandry. Affine determinant programs: A framework for obfuscation and witness encryption. In Thomas Vidick, editor, *ITCS 2020*, volume 151, pages 82:1–82:39. LIPIcs, January 2020.

[BJK15] Allison Bishop, Abhishek Jain, and Lucas Kowalczyk. Function-hiding inner product encryption. In Tetsu Iwata and Jung Hee Cheon, editors, *ASIACRYPT 2015, Part I*, volume 9452 of *LNCS*, pages 470–491. Springer, Heidelberg, November / December 2015.

[BKKV10] Zvika Brakerski, Yael Tauman Kalai, Jonathan Katz, and Vinod Vaikuntanathan. Overcoming the hole in the bucket: Public-key cryptography resilient to continual memory leakage. In *51st FOCS*, pages 501–510. IEEE Computer Society Press, October 2010.

[BKM$^+$19] Allison Bishop, Lucas Kowalczyk, Tal Malkin, Valerio Pastro, Mariana Raykova, and Kevin Shi. In pursuit of clarity in obfuscation. *IACR Cryptol. ePrint Arch.*, 2019:463, 2019.

[BLMR13] Dan Boneh, Kevin Lewi, Hart William Montgomery, and Ananth Raghunathan. Key homomorphic PRFs and their applications. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 410–428. Springer, Heidelberg, August 2013.

[BLMZ19] James Bartusek, Tancrède Lepoint, Fermi Ma, and Mark Zhandry. New techniques for obfuscating conjunctions. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part III*, volume 11478 of *LNCS*, pages 636–666. Springer, Heidelberg, May 2019.

[BMSZ16] Saikrishna Badrinarayanan, Eric Miles, Amit Sahai, and Mark Zhandry. Post-zeroizing obfuscation: New mathematical tools, and the case of evasive circuits. In *Advances in Cryptology - EUROCRYPT*, pages 764–791, 2016.

[BNPW16] Nir Bitansky, Ryo Nishimaki, Alain Passelègue, and Daniel Wichs. From cryptomania to obfustopia through secret-key functional encryption. Cryptology ePrint Archive, Report 2016/558, 2016. `http://eprint.iacr.org/2016/558`.

[BPR15] Nir Bitansky, Omer Paneth, and Alon Rosen. On the cryptographic hardness of finding a Nash equilibrium. In Venkatesan Guruswami, editor, *56th FOCS*, pages 1480–1498. IEEE Computer Society Press, October 2015.

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

[BR14] Zvika Brakerski and Guy N. Rothblum. Virtual black-box obfuscation for all circuits via generic graded encoding. In *TCC*, pages 1–25, 2014.

[BV11] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In Rafail Ostrovsky, editor, *52nd FOCS*, pages 97–106. IEEE Computer Society Press, October 2011.

[BV15a] Nir Bitansky and Vinod Vaikuntanathan. Indistinguishability obfuscation from functional encryption. In *FOCS*. IEEE, 2015.

[BV15b] Zvika Brakerski and Vinod Vaikuntanathan. Constrained key-homomorphic PRFs from standard lattice assumptions - or: How to secretly embed a circuit in your PRF. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015, Part II*, volume 9015 of *LNCS*, pages 1–30. Springer, Heidelberg, March 2015.

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

[CCL18] Yi-Hsiu Chen, Kai-Min Chung, and Jyun-Jie Liao. On the complexity of simulating auxiliary input. In *EUROCRYPT*, Cham, 2018.

[CDM$^+$18] Geoffroy Couteau, Aurélien Dupin, Pierrick Méaux, Mélissa Rossi, and Yann Rotella. On the concrete security of Goldreich's pseudorandom generator. In Thomas Peyrin and Steven Galbraith, editors, *ASIACRYPT 2018, Part II*, volume 11273 of *LNCS*, pages 96–124. Springer, Heidelberg, December 2018.

[CGH$^+$15] Jean-Sébastien Coron, Craig Gentry, Shai Halevi, Tancrède Lepoint, Hemanta K. Maji, Eric Miles, Mariana Raykova, Amit Sahai, and Mehdi Tibouchi. Zeroizing without low-level zeroes: New MMAP attacks and their limitations. In *CRYPTO*, 2015.

[CHL$^+$15] Jung Hee Cheon, Kyoohyung Han, Changmin Lee, Hansol Ryu, and Damien Stehlé. Cryptanalysis of the multilinear map over the integers. In *EUROCRYPT*, 2015.

[CHN$^+$16] Aloni Cohen, Justin Holmgren, Ryo Nishimaki, Vinod Vaikuntanathan, and Daniel Wichs. Watermarking cryptographic capabilities. In *STOC*, 2016.

[CLL$^+$12] Jie Chen, Hoon Wei Lim, San Ling, Huaxiong Wang, and Hoeteck Wee. Shorter IBE and signatures via asymmetric pairings. In Michel Abdalla and Tanja Lange, editors, *Pairing-Based Cryptography - Pairing 2012 - 5th International Conference, Cologne, Germany, May 16-18, 2012, Revised Selected Papers*, volume 7708 of *Lecture Notes in Computer Science*, pages 122–140. Springer, 2012.

[CLR15] Jung Hee Cheon, Changmin Lee, and Hansol Ryu. Cryptanalysis of the new clt multilinear maps. Cryptology ePrint Archive, Report 2015/934, 2015. `http://eprint.iacr.org/`.

[CLT13] Jean-Sébastien Coron, Tancrède Lepoint, and Mehdi Tibouchi. Practical multilinear maps over the integers. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 476–493. Springer, Heidelberg, August 2013.

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

[CM01] Mary Cryan and Peter Bro Miltersen. On pseudorandom generators in NC. In Jiri Sgall, Ales Pultr, and Petr Kolman, editors, *Mathematical Foundations of Computer Science 2001, 26th International Symposium, MFCS 2001 Marianske Lazne, Czech Republic, August 27-31, 2001, Proceedings*, volume 2136 of *Lecture Notes in Computer Science*, pages 272–284. Springer, 2001.

[DGG$^+$16] Nico Döttling, Sanjam Garg, Divya Gupta, Peihan Miao, and Pratyay Mukherjee. Obfuscation from low noise multilinear maps. *IACR Cryptology ePrint Archive*, 2016:599, 2016.

[DGN$^+$17] Nico Döttling, Satrajit Ghosh, Jesper Buus Nielsen, Tobias Nilges, and Roberto Trifiletti. TinyOLE: Efficient actively secure two-party computation from oblivious linear function evaluation. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017*, pages 2263–2276. ACM Press, October / November 2017.

[DHRW16] Yevgeniy Dodis, Shai Halevi, Ron D. Rothblum, and Daniel Wichs. Spooky encryption and its applications. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part III*, volume 9816 of *LNCS*, pages 93–122. Springer, Heidelberg, August 2016.

[GGG$^+$14] Shafi Goldwasser, S. Dov Gordon, Vipul Goyal, Abhishek Jain, Jonathan Katz, Feng-Hao Liu, Amit Sahai, Elaine Shi, and Hong-Sheng Zhou. Multi-input functional encryption. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 578–602. Springer, Heidelberg, May 2014.

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

[GGH$^+$13b] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *54th FOCS*, pages 40–49. IEEE Computer Society Press, October 2013.

[GGH15] Craig Gentry, Sergey Gorbunov, and Shai Halevi. Graph-induced multilinear maps from lattices. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015, Part II*, volume 9015 of *LNCS*, pages 498–527. Springer, Heidelberg, March 2015.

[Gil52] E. N. Gilbert. A comparison of signalling alphabets. *The Bell System Technical Journal*, 31(3):504–522, 1952.

[GJK18] Craig Gentry, Charanjit S. Jutla, and Daniel Kane. Obfuscation using tensor products. *Electronic Colloquium on Computational Complexity (ECCC)*, 25:149, 2018.

[GJLS20] Romain Gay, Aayush Jain, Huijia Lin, and Amit Sahai. Indistinguishability obfuscation from simple-to-state hard problems: New assumptions, new techniques, and simplification. *IACR Cryptol. ePrint Arch.*, 2020:764, 2020.

[GKP$^+$13] Shafi Goldwasser, Yael Tauman Kalai, Raluca A. Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich. Reusable garbled circuits and succinct functional encryption. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 555–564. ACM, 2013.

[GKR08] Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. One-time programs. In David Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 39–56. Springer, Heidelberg, August 2008.

[GKW17] Rishab Goyal, Venkata Koppula, and Brent Waters. Lockable obfuscation. In Chris Umans, editor, *58th FOCS*, pages 612–621. IEEE Computer Society Press, October 2017.

[GNN17] Satrajit Ghosh, Jesper Buus Nielsen, and Tobias Nilges. Maliciously secure oblivious linear function evaluation with constant overhead. In Tsuyoshi Takagi and Thomas Peyrin, editors, *ASIACRYPT 2017, Part I*, volume 10624 of *LNCS*, pages 629–659. Springer, Heidelberg, December 2017.

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

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

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In Richard E. Ladner and Cynthia Dwork, editors, *40th ACM STOC*, pages 197–206. ACM Press, May 2008.

[GR04] Steven D. Galbraith and Victor Rotger. Easy decision-diffie-hellman groups. *IACR Cryptol. ePrint Arch.*, 2004:70, 2004.

[GS08] Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. In Nigel P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 415–432. Springer, Heidelberg, April 2008.

[GS16] Sanjam Garg and Akshayaram Srinivasan. Single-key to multi-key functional encryption with polynomial loss. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B, Part II*, volume 9986 of *LNCS*, pages 419–442. Springer, Heidelberg, October / November 2016.

[GSW13] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 75–92. Springer, Heidelberg, August 2013.

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

[GVW13] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Attribute-based encryption for circuits. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th ACM STOC*, pages 545–554. ACM Press, June 2013.

[GVW15] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Predicate encryption for circuits from LWE. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 503–523. Springer, Heidelberg, August 2015.

[Hal15] Shai Halevi. Graded encoding, variations on a scheme. *IACR Cryptology ePrint Archive*, 2015:866, 2015.

[HB01] Nicholas J. Hopper and Manuel Blum. Secure human identification protocols. In Colin Boyd, editor, *ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 52–66. Springer, Heidelberg, December 2001.

[HJ15] Yupu Hu and Huiwen Jia. Cryptanalysis of GGH map. *IACR Cryptology ePrint Archive*, 2015:301, 2015.

[HJK$^+$16] Dennis Hofheinz, Tibor Jager, Dakshita Khurana, Amit Sahai, Brent Waters, and Mark Zhandry. How to generate and use universal samplers. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part II*, volume 10032 of *LNCS*, pages 715–744. Springer, Heidelberg, December 2016.

[HSW13] Susan Hohenberger, Amit Sahai, and Brent Waters. Full domain hash from (leveled) multilinear maps and identity-based aggregate signatures. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 494–512. Springer, Heidelberg, August 2013.

[IPS09] Yuval Ishai, Manoj Prabhakaran, and Amit Sahai. Secure arithmetic computation with no honest majority. In *TCC Conference, TCC 2009, San Francisco, CA, USA, March 15-17, 2009. Proceedings*, pages 294–314, 2009.

[JKMS20] Aayush Jain, Alexis Korb, Nathan Manohar, and Amit Sahai. Amplifying functional encryption, unconditionally. *CRYPTO*, 2020, 2020.

[JLMS19] Aayush Jain, Huijia Lin, Christian Matt, and Amit Sahai. How to leverage hardness of constant-degree expanding polynomials overa $\mathbb{R}$ to build $i\mathcal{O}$. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part I*, volume 11476 of *LNCS*, pages 251–281. Springer, Heidelberg, May 2019.

[JLS19] Aayush Jain, Huijia Lin, and Amit Sahai. Simplifying constructions and assumptions for $i\mathcal{O}$. *IACR Cryptol. ePrint Arch.*, 2019:1252, 2019.

[JR13] Charanjit S. Jutla and Arnab Roy. Shorter quasi-adaptive NIZK proofs for linear subspaces. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT 2013, Part I*, volume 8269 of *LNCS*, pages 1–20. Springer, Heidelberg, December 2013.

[KLW15] Venkata Koppula, Allison Bishop Lewko, and Brent Waters. Indistinguishability obfuscation for turing machines with unbounded memory. In *STOC*, 2015.

[KMOW17] Pravesh K. Kothari, Ryuhei Mori, Ryan O'Donnell, and David Witmer. Sum of squares lower bounds for refuting any CSP. In Hamed Hatami, Pierre McKenzie, and Valerie King, editors, *49th ACM STOC*, pages 132–145. ACM Press, June 2017.

[KNT18] Fuyuki Kitagawa, Ryo Nishimaki, and Keisuke Tanaka. Obfustopia built on secret-key functional encryption. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part II*, volume 10821 of *LNCS*, pages 603–648. Springer, Heidelberg, April / May 2018.

[Lin16] Huijia Lin. Indistinguishability obfuscation from constant-degree graded encoding schemes. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part I*, volume 9665 of *LNCS*, pages 28–57. Springer, Heidelberg, May 2016.

[Lin17] Huijia Lin. Indistinguishability obfuscation from SXDH on 5-linear maps and locality-5 PRGs. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part I*, volume 10401 of *LNCS*, pages 599–629. Springer, Heidelberg, August 2017.

[LM16] Baiyu Li and Daniele Micciancio. Compactness vs collusion resistance in functional encryption. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B, Part II*, volume 9986 of *LNCS*, pages 443–468. Springer, Heidelberg, October / November 2016.

[LM18] Huijia Lin and Christian Matt. Pseudo flawed-smudging generators and their application to indistinguishability obfuscation. *IACR Cryptology ePrint Archive*, 2018:646, 2018.

[LPST16] Huijia Lin, Rafael Pass, Karn Seth, and Sidharth Telang. Indistinguishability obfuscation with non-trivial efficiency. In *IACR International Workshop on Public Key Cryptography*, pages 447–462. Springer, 2016.

[LT17] Huijia Lin and Stefano Tessaro. Indistinguishability obfuscation from trilinear maps and block-wise local PRGs. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part I*, volume 10401 of *LNCS*, pages 630–660. Springer, Heidelberg, August 2017.

[LV16] Huijia Lin and Vinod Vaikuntanathan. Indistinguishability obfuscation from DDH-like assumptions on constant-degree graded encodings. In Irit Dinur, editor, *57th FOCS*, pages 11–20. IEEE Computer Society Press, October 2016.

[LV17] Alex Lombardi and Vinod Vaikuntanathan. Limits on the locality of pseudorandom generators and applications to indistinguishability obfuscation. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017, Part I*, volume 10677 of *LNCS*, pages 119–137. Springer, Heidelberg, November 2017.

[MF15] Brice Minaud and Pierre-Alain Fouque. Cryptanalysis of the new multilinear map over the integers. Cryptology ePrint Archive, Report 2015/941, 2015. `http://eprint.iacr.org/`.

[MP13] Daniele Micciancio and Chris Peikert. Hardness of SIS and LWE with small parameters. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 21–39. Springer, Heidelberg, August 2013.

[MR04] Daniele Micciancio and Oded Regev. Worst-case to average-case reductions based on Gaussian measures. In *45th FOCS*, pages 372–381. IEEE Computer Society Press, October 2004.

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

[MSZ16] Eric Miles, Amit Sahai, and Mark Zhandry. Annihilation attacks for multilinear maps: Cryptanalysis of indistinguishability obfuscation over GGH13. In *Advances in Cryptology - CRYPTO*, 2016.

[MW16] Pratyay Mukherjee and Daniel Wichs. Two round multiparty computation via multi-key FHE. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 735–763. Springer, Heidelberg, May 2016.

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

[Pei09] Chris Peikert. Public-key cryptosystems from the worst-case shortest vector problem: extended abstract. In Michael Mitzenmacher, editor, *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*, pages 333–342. ACM, 2009.

[PST14] Rafael Pass, Karn Seth, and Sidharth Telang. Indistinguishability obfuscation from semantically-secure multilinear encodings. In *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part I*, pages 500–517, 2014.

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In *STOC*, pages 84–93, 2005.

[SW14] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In David B. Shmoys, editor, *STOC*, pages 475–484. ACM, 2014.

[Var57] Rom Varshamov. Estimate of the number of signals in error correcting codes. *Dokl. Akad. Nauk SSSR*, 1957.

[Ver01] Eric R. Verheul. Evidence that XTR is more secure than supersingular elliptic curve cryptosystems. In Birgit Pfitzmann, editor, *EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 195–210. Springer, Heidelberg, May 2001.

[WZ17] Daniel Wichs and Giorgos Zirdelis. Obfuscating compute-and-compare programs under LWE. In Chris Umans, editor, *58th FOCS*, pages 600–611. IEEE Computer Society Press, October 2017.

A Partially Hiding Functional Encryption

We recall the notion of Partially-hiding Functional Encryption (PHFE) schemes; some of the text in this section is taken verbatim from [GJLS20]. PHFE involves functional secret keys, each of which is associated with some 2-ary function f, and decryption of a ciphertext encrypting $(\boldsymbol{x}, \boldsymbol{y})$ with such a key reveals $f(\boldsymbol{x}, \boldsymbol{y})$, $\boldsymbol{x}$, f, and nothing more about $\boldsymbol{y}$. Since only the input $\boldsymbol{y}$ is hidden, such an FE scheme is called partially-hiding FE. FE can be viewed as a special case of PHFE where the public input is the empty string. The notion was originally introduced by [GVW12] and a similar notion of partially-hiding predicate encryption was proposed and constructed by [GVW15].

We denote functionality by $\mathcal{F} : \mathcal{X} \times \mathcal{Y} \to \mathcal{Z}$. The functionality ensemble $\mathcal{F}$ as well as the message ensembles $\mathcal{X}$ and $\mathcal{Y}$ are indexed by two parameters: n and λ (for example $\mathcal{F}_{n,\lambda}$), where λ is the security parameter and n is a length parameter and can be viewed as a function of λ.

Definition A.1. *(Syntax of a PHFE/FE Scheme.) A* secret key *partially hiding functional encryption scheme,* PHFE, *for the functionality $\mathcal{F} : \mathcal{X} \times \mathcal{Y} \to \mathcal{Z}$ consists of the following polynomial time algorithms:*

- $\mathsf{PPGen}(1^\lambda, 1^n)$: *The public parameter generation algorithm is a randomized algorithm that takes as input n and λ and outputs a string* crs*.*

- $\mathsf{Setup}(\mathsf{crs})$*: The setup algorithm is a randomized algorithm that on input* crs*, returns a master secret key* msk*.*

- $\mathsf{Enc}(\mathsf{msk}, (x, y) \in \mathcal{X}_{n,\lambda} \times \mathcal{Y}_{n,\lambda})$*: The encryption algorithm is a randomized algorithm that takes in a master secret key and a message (x, y) and returns the ciphertext* ct *along with the input x. x is referred to as the public input whereas y is called the private input.*

- $\mathsf{KeyGen}(\mathsf{msk}, f \in \mathcal{F}_{n,\lambda})$*: The key generation algorithm is a randomized algorithms that takes in a description of a function $f \in \mathcal{F}_{n,\lambda}$ and returns* sk_f*, a decryption key for f.*

- $\mathsf{Dec}(\mathsf{sk}_f, (x, \mathsf{ct}))$*: The decryption algorithm is a deterministic algorithm that returns a value z in $\mathcal{Z}$, or $\perp$ if it fails.*

A functional encryption scheme is a partially hiding functional encryption scheme, where $\mathcal{X}_{n,\lambda} = \emptyset$ for all n, λ.

Define three levels of efficiency: let $S = S(\lambda, n)$ be the maximum size of functions in $\mathcal{F}_{\lambda,n}$; ciphertext ct *produced by running* $\mathsf{PPGen}, \mathsf{Setup}, \mathsf{Enc}$ *honestly as above has the following sizes with respect to some arbitrary constant $\epsilon \in (0, 1]$.*

- *Sublinear compactness:* $\mathsf{poly}(\lambda, n)S^{1-\epsilon}$
- *Sublinear compactness and linear dependency on input length:* $\mathsf{poly}(\lambda)(n + S^{1-\epsilon})$
- *Linear Efficiency:* $\mathsf{poly}(\lambda)n$

We surpress the public input in notation in the case of functional encryption.

Definition A.2. *(Correctness of a PHFE/FE scheme.) A secret key partially hiding functional encryption scheme,* PHFE, *for the functionality $\mathcal{F} : \mathcal{X} \times \mathcal{Y} \to \mathcal{Z}$ is correct if for every $\lambda \in \mathbb{N}$ and every polynomial $n(\lambda) \in \mathbb{N}$, for every $(x, y) \in \mathcal{X}_{n,\lambda} \times \mathcal{Y}_{n,\lambda}$ and every $f \in \mathcal{F}_{n,\lambda}$, we have:*

$$\Pr\left[\mathsf{Dec}(\mathsf{sk}_f, x, \mathsf{ct})) = f(x, y) \;\middle|\; \begin{array}{l} \mathsf{PPGen}(1^\lambda, 1^n) \to \mathsf{crs} \\ \mathsf{Setup}(\mathsf{crs}) \to \mathsf{msk} \\ \mathsf{Enc}(\mathsf{msk}, (x, y)) \to (x, \mathsf{ct}) \\ \mathsf{KeyGen}(\mathsf{msk}, f) \to \mathsf{sk}_f \end{array}\right] = 1$$

Definition A.3 (Simulation security)**.** *A secret-key partially hiding functional encryption scheme* PHFE *for functionality $\mathcal{F} : \mathcal{X} \times \mathcal{Y} \to \mathcal{Z}$ is (weakly selective) (T, ϵ)-SIM secure, if for every positive polynomials $n = n(\lambda)$, $Q_{\mathsf{ct}} = Q_{\mathsf{ct}}(\lambda)$, $Q_{\mathsf{sk}} = Q_{\mathsf{sk}}(\lambda)$, ensembles $\{(x, y)\}$, $\{\{(x_i, y_i)\}_{i \in [Q_{\mathsf{ct}}]}\}$ in $\mathcal{X}_{\lambda,n} \times \mathcal{Y}_{\lambda,n}$ and $\{\{f_j\}_{j \in [Q_{\mathsf{sk}}]}\}$ in $\mathcal{F}_{\lambda,n}$, the following distributions are (T, ϵ)-indistinguishable.*

$$\left\{ \left(\mathsf{crs},\ \mathsf{ct},\ \{\mathsf{ct}_i\}_{i \in [Q_{\mathsf{ct}}]},\ \{\mathsf{sk}_j\}_{j \in [Q_{\mathsf{sk}}]}\right) \;\middle|\; \begin{array}{l} \mathsf{crs} \leftarrow \mathsf{PPGen}(1^\lambda, 1^n),\ \mathsf{msk} \leftarrow \mathsf{Setup}(\mathsf{crs}) \\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{msk}, (x, y)) \\ \forall i \in [Q_{\mathsf{ct}}],\ \mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{msk}, (x_i, y_i)) \\ \forall j \in [Q_{\mathsf{sk}}],\ \mathsf{sk}_j \leftarrow \mathsf{KeyGen}(\mathsf{msk}, f_j) \end{array} \right\}$$

$$\left\{ \left(\mathsf{crs},\ \widetilde{\mathsf{ct}},\ \{\widetilde{\mathsf{ct}}_i\}_{i \in [Q_{\mathsf{ct}}]},\ \{\widetilde{\mathsf{sk}}_j\}_{j \in [Q_{\mathsf{sk}}]}\right) \;\middle|\; \begin{array}{l} \mathsf{crs} \leftarrow \mathsf{PPGen}(1^\lambda, 1^n),\ \widetilde{\mathsf{msk}} \leftarrow \widetilde{\mathsf{Setup}}(\mathsf{crs}) \\ \widetilde{\mathsf{ct}} \leftarrow \widetilde{\mathsf{Enc}}_1(\widetilde{\mathsf{msk}}, x) \\ \forall i \in [Q_{\mathsf{ct}}],\ \widetilde{\mathsf{ct}}_i \leftarrow \widetilde{\mathsf{Enc}}_2(\widetilde{\mathsf{msk}}, (x_i, y_i)) \\ \forall j \in [Q_{\mathsf{sk}}],\ \widetilde{\mathsf{sk}}_j \leftarrow \mathsf{KeyGen}(\widetilde{\mathsf{msk}}, f_j, f_j(x, y)) \end{array} \right\}$$

Definition A.4 (Indistinguishability security)**.** *A secret-key functional encryption scheme* FE *for functionality $\mathcal{F} : \mathcal{X} \to \mathcal{Z}$ is (weakly selective) (T, ϵ)-IND secure, if for every positive polynomials $n = n(\lambda)$, $Q_{\mathsf{ct}} = Q_{\mathsf{ct}}(\lambda)$, $Q_{\mathsf{sk}} = Q_{\mathsf{sk}}(\lambda)$, ensembles $\{\{x_{i,0}, x_{i,0}\}_{i \in [Q_{\mathsf{ct}}]}\}$ in $\mathcal{X}_{\lambda,n}$ and $\{\{f_j\}_{j \in [Q_{\mathsf{sk}}]}\}$ in $\mathcal{F}_{\lambda,n}$, the following distributions for $b \in \{0, 1\}$ are (T, ϵ)-indistinguishable.*

$$\left\{ \left(\mathsf{crs},\ \{\mathsf{ct}_i\}_{i \in [Q_{\mathsf{ct}}]},\ \{\mathsf{sk}_j\}_{j \in [Q_{\mathsf{sk}}]}\right) \;\middle|\; \begin{array}{l} \mathsf{crs} \leftarrow \mathsf{PPGen}(1^\lambda, 1^n),\ \mathsf{msk} \leftarrow \mathsf{Setup}(\mathsf{crs}) \\ \forall i \in [Q_{\mathsf{ct}}],\ \mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{msk}, x_{i,b}) \\ \forall j \in [Q_{\mathsf{sk}}],\ \mathsf{sk}_j \leftarrow \mathsf{KeyGen}(\mathsf{msk}, f_j) \end{array} \right\}$$

B Recap of constant-depth functional encryption

We give a self-contained description of a construction of *1-key* secret-key FE for NC^0 satisfying *sublinear compactness with linear dependency on input length*, which can be transformed to $i\mathcal{O}$ as described in Section 5. We emphasize that the construction of FE for NC^0 recalled here was given by prior works [AJL+19, JLMS19, LV16, Lin16]. The purpose of this appendix is providing a clean and self-contained description of the construction for convenient lookup, and we omit the security proof.

Consider the class of NC^0 functions $g : \{0,1\}^l \to \{0,1\}^m$. Such functions can be computed by a multilinear polynomial with 1/-1 coefficient of some constant degree D. We now describe the FE scheme for computing such functions, which uses the following ingredients.

Ingredients. Let λ be the security parameter and $p = p(\lambda) = O(2^\lambda)$ an efficiently computable prime modulus.

- LWE over $\mathbb{Z}_p$ with subexponential modulus to noise ratio 2^{k^ϵ} where k is the dimension of LWE secret and ϵ is some arbitrary constant in $(0,1)$.

  Related parameters are set to:

  - We use polynomially large noises: Let $\chi_{\alpha,B}$ be the truncated discrete gaussian distribution with parameter α and support $[-B,B] \cap \mathbb{Z}$, where $\alpha \leq B$ are set appropriately and of magnitude $\mathsf{poly}(\lambda)$. As such, the modulus-to-noise ratio is $p/\mathsf{poly}(\lambda)$.
  - Set the LWE dimension k appropriately $k = \Theta(\lambda^{1/\epsilon})$ such that the modulus-to-noise ratio $p/\mathsf{poly}(\lambda)$ is upper bounded by 2^{k^ϵ}.

  We will use the basic homomorphic encryption scheme by [BV11] based on LWE. An encryption of a Boolean string $\boldsymbol{x}$ has form $\boldsymbol{A}, \boldsymbol{b} = \boldsymbol{sA} + 2\boldsymbol{e} + \boldsymbol{x}$ over $\mathbb{Z}_p$ and supports homomorphic evaluation of constant degree polynomials over $\mathbb{Z}_p$ (without relinearization).

- A perturbation resilient generator $\Delta\mathsf{RG} = (\mathsf{SetupPoly}, \mathsf{SetupSeed}, \mathsf{Eval})$ with stretch $\tau > 1$ and complexity $(\mathsf{arith\text{-}NC}^1, \deg\ 2)$ over $\mathbb{Z}_p$. Such a $\Delta\mathsf{RG}$ was constructed in Section 5, based on Boolean PRGs in NC^0 the LPN assumption over $\mathbb{Z}_p$.

  Related parameters are set to:

  - The bound on the noises to be smudged is set to be $B^D \cdot l^D \cdot \lambda$.
  - The output length of $\Delta\mathsf{RG}$ is m, matching the output length of the NC^0 computation.
  - The seed length is then $n\,\mathsf{poly}(\lambda)$ for $n = m^{1/\tau}$.

- A SIM-secure collusion-resistant secret-key scheme for ($\mathsf{arith\text{-}NC}^1$, deg 2), $\mathsf{PHFE} =$ ($\mathsf{PHFE.PPGen}$, $\mathsf{PHFE.Setup}$, $\mathsf{PHFE.Enc}$, $\mathsf{PHFE.KeyGen}$, $\mathsf{PHFE.Dec}$). This can be built from the SXDH assumption over asymmetric bilinear groups of order p as presented in [JLS19].

  Related parameters are set to:

  - The input length parameter n' is an efficiently computable function depending on n, k, D set implicitly in the Enc algorithm below.

Construction: The NC^0-FE scheme $\mathsf{FE} = (\mathsf{PPGen}, \mathsf{Setup}, \mathsf{Enc}, \mathsf{KeyGen}, \mathsf{Dec})$ is as follows:

$\mathsf{crs} \leftarrow \mathsf{PPGen}(1^\lambda, 1^l)$**:** Sample $\boldsymbol{A} \leftarrow \mathbb{Z}_p^{k\times l}$, $\mathsf{crs}_{\mathsf{PHFE}} \leftarrow \mathsf{PHFE.PPGen}(1^\lambda, 1^{n'})$, and $I \leftarrow \Delta\mathsf{RG.SetupPoly}(1^\lambda, 1^n, 1^{B^D\cdot l^D\cdot\lambda})$. Output $\mathsf{crs} = (\mathsf{crs}_{\mathsf{PHFE}}, I, \boldsymbol{A})$.

$\mathsf{msk} \leftarrow \mathsf{Setup}(\mathsf{crs})$**:** Sample $\mathsf{msk}_{\mathsf{PHFE}} \leftarrow \mathsf{PHFE.Setup}(\mathsf{crs}_{\mathsf{PHFE}})$ and output $\mathsf{msk} = (\mathsf{msk}_{\mathsf{PHFE}}, \mathsf{crs})$.

$\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{msk}, \boldsymbol{x} \in \{0,1\}^l)$**:**

- Sample $(P, S) \leftarrow \Delta\mathsf{RG.SetupSeed}(I)$. Note that the seed has length $|P| + |S| = n\,\mathsf{poly}(\lambda)$.
- Encrypt $\boldsymbol{x}$ as follows: Sample a secret $\boldsymbol{s} \leftarrow \mathbb{Z}_p^k$ and noise vector $\boldsymbol{e} \leftarrow \chi_{\alpha,B}^l$, and compute $\boldsymbol{b} = \boldsymbol{s}\boldsymbol{A} + 2\boldsymbol{e} + \boldsymbol{x}$.
- Let $\overline{\boldsymbol{s}} = (1\|\boldsymbol{s})$ and compute $\overline{\boldsymbol{s}}^{\otimes\lceil\frac{D}{2}\rceil}$.
- Set public input $X = (P, \boldsymbol{b})$ and private input $Y = (S, \overline{\boldsymbol{s}}^{\otimes\lceil\frac{D}{2}\rceil})$, and encrypt them using PHFE, $\mathsf{ct} \leftarrow \mathsf{PHFE.Enc}(\mathsf{msk}, (X, Y))$.

Output ct.

$\mathsf{sk} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, g)$**:** Output a PHFE key $\mathsf{sk}_{\mathsf{PHFE}} \leftarrow \mathsf{PHFE.KeyGen}(\mathsf{msk}, G)$ for the following function G.

Function G takes public input X and private input Y and does the following:

- Compute $f(\boldsymbol{x}) + 2\boldsymbol{e}'$ via a polynomial $G^{(1)}$ that has degree D in X and degree 2 in Y.

  Function $G^{(1)}$ is defined as follows: Since f is a degree D multilinear polynomial with 1/-1 coefficients, we have (using the same notation as in Section 4)

  $$\forall j \in [m],\ f_j(\boldsymbol{x}) = L_j((x_v)_{v\in f_j}) \text{ for some linear } L_j \text{ with 1/-1 coefficients .}$$

  The decryption equation for $\boldsymbol{b}$ is

  $$\forall i \in [l],\ x_i + 2e_i = \langle \boldsymbol{c}_i,\ \overline{\boldsymbol{s}}\rangle \qquad \boldsymbol{c}_i = -\boldsymbol{a}_i^{\mathrm{T}}||b_i,\ \boldsymbol{a}_i \text{ is the } i\text{th column of } \boldsymbol{A} \text{ .}$$

  Thus, we have

  $$\begin{aligned} \forall \text{ degree } D \text{ monomial } v,\ x_v + 2e_v &= \langle \otimes_{i\in v}\boldsymbol{c}_i,\ \otimes_{i\in v}\overline{\boldsymbol{s}}\rangle \\ \forall j \in [m],\ f_j(\boldsymbol{x}) + 2e'_j &= L_j\left((\langle \otimes_{i\in v}\boldsymbol{c}_i,\ \otimes_{i\in v}\overline{\boldsymbol{s}}\rangle)_{v\in f_j}\right) \\ e'_j &= L_j((e_v)_{v\in f_j}) \text{ has } \mathsf{poly}(\lambda) \text{ magnitude} \end{aligned}$$

  Define $G^{(1)}$ to be the polynomial that computes $f(\boldsymbol{x}) + 2\boldsymbol{e}'$

  $$G^{(1)}(X, Y) = f(\boldsymbol{x}) + 2\boldsymbol{e}' \text{ ,}$$

  with degree D in X (containing $\boldsymbol{b}$) and degree 2 in Y (containing $\overline{\boldsymbol{s}}^{\otimes\lceil\frac{D}{2}\rceil}$). $G^{(1)}$ also depends on $\boldsymbol{A}$.

- Compute $\boldsymbol{r} \leftarrow \Delta\mathsf{RG.Eval}(I, \mathsf{sd})$.
- Output $\boldsymbol{y}' = \boldsymbol{y} + 2\boldsymbol{e}_f + 2\boldsymbol{r}$.

Observe that because of the complexity of $G^{(1)}$ and $\Delta\mathsf{RG}$, G is in $(\mathsf{arith}\text{-}\mathsf{NC}^1, \deg 2)$.

$\mathsf{Dec}(\mathsf{sk}, \mathsf{ct})$: Decrypt the PHFE ciphertext $\boldsymbol{y} + 2\boldsymbol{e}' = G(X, Y) \leftarrow \mathsf{PHFE.Dec}(\mathsf{sk}_{\mathsf{PHFE}}, \mathsf{ct}_{\mathsf{PHFE}})$, which reveals $\boldsymbol{y} \bmod 2$.

More precisely, the decryption of PHFE built from bilinear groups produces $g_T^{(y_j+2e'_j)}$ for every $j \in [m]$, where g_T is the generator of the target group. Thus, decryption needs to first extracts $y_j + 2e'_j$ by brute force discrete logarithm, which is efficient as e'_j has $\mathsf{poly}(\lambda)$ magnitude.

Sublinear Compactness with Linear Dependency on Input Length Observe that the ciphertext ct produced above has size $\mathsf{poly}(\lambda, l)S^{1-\epsilon} = \mathsf{poly}(\lambda, l)m^{1-\epsilon}$ for some $\epsilon \in (0, 1)$, following from the following facts:

- By the linear efficiency of PHFE, $|\mathsf{ct}| = \mathsf{poly}(\lambda)(|X| + |Y|)$.
- The seed P, S of $\Delta\mathsf{RG}$ has length $m^{1/\tau}$ for $\tau > 1$.
- $|\boldsymbol{b}| = k \log p = O(k\lambda)$.
- $\overline{\boldsymbol{s}}^{\otimes \lceil \frac{D}{2} \rceil}$ has size $k^{\lceil \frac{D}{2} \rceil} \log p = O(\lambda^{(\lceil \frac{D}{2} \rceil / \epsilon) + 1}) = \mathsf{poly}(\lambda)$.

ECCC ISSN 1433-8092

https://eccc.weizmann.ac.il