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.

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:

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

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 $\mathbf{M} \in \mathbb{Z}_q^{n_1\times n_2}$, we denote by $|\mathbf{M}|$ the bit length of $\mathbf{M}$. In this case, $|\mathbf{M}| = n_1 \cdot n_2 \cdot \log_2 q$. We also overload this operator in that, for any set S, we use $|S|$ to denote the cardinality of S. The meaning should be inferred from context.

For any two polynomials $a(\lambda, n), b(\lambda, n) : \mathbb{N} \times \mathbb{N} \to \mathbb{R}^{\geq 0}$, we say that a is polynomially smaller than b, denoted as $a \ll b$, if there exists an $\epsilon \in (0, 1)$ and a constant $c > 0$ such that $a < b^{1-\epsilon} \cdot \lambda^c$ for all large enough $n, \lambda \in \mathbb{N}$. 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 $\lambda \in \mathbb{N}$,*

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

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*

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

- (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 $\{0,1\}^n \times \{0,1\}^\ell \to \{0,1\}$ 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 $\{0,1\}^n \times \{0,1\}^{s'} \to \{0,1\}^\ell$. 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,* $\mathsf{sPRG} \in (\mathbb{C},\ \deg d)$*, 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 arith-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* (arith-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:

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

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

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

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 $\mathsf{Corr} = \mathbf{UV}$, and compute the correct output as follows:

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

where (k_j, l_j) is the corresponding index of the output bit j, in the $\sqrt{m} \times \sqrt{m}$ matrix. When $z \ll \sqrt{m}$, the matrices $\mathbf{U}, \mathbf{V}$ have $2z\sqrt{m}$ field elements, which is polynomially smaller than $m = n^\tau$. As such, $G^{(2)}$ is expanding.

Moreover, observe that $G^{(2)}$ has only degree 2 in the private seed and is completely oblivious of where the bad outputs are.

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:

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

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:

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

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

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

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:* $\exists i \in [B]$, $|\phi^{-1}_{\mathsf{bkt}}(i)| > c$,
  3. **or** *Some bucket contains too many* bad *outputs:* $\exists i \in [B]$, $|\phi^{-1}_{\mathsf{bkt}}(i) \cap \mathsf{BAD}| > t$.

  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:

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

  (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 $\mathsf{Eval}(\boldsymbol{\sigma} + \boldsymbol{e})$:

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

  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:

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

  We have that $G^{(2)}$ agrees with the correct output $\boldsymbol{y} = \mathsf{Eval}(\boldsymbol{\sigma})$ if $\mathsf{flag} = 1$. This is because under the three conditions for $\mathsf{flag} = 1$, every entry j in the correction vector Corr_j is placed at entry $\left(\mathbf{M}_{\phi_{\mathsf{bkt}}(j)}\right)_{\phi_{\mathsf{ind}}(j)}$. Adding it back as above produces the correct output.

  Observe that the function is quadratic in S and degree d in the public component of the seed P.

- When $\mathsf{flag} = 0$, however, sPRG needs to output all zero. This can be done by simply multiplying flag to the output of $G^{(2)}$, giving the final polynomial

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

  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.

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

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 $\mathsf{flag} \cdot \boldsymbol{y}$.

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

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

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

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

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

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

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

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:

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

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* PRG*s 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) (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 (ΔRG) [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 ΔRG has polynomial stretch if above $\tau > 1$ for some constant τ.

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

Definition 5.2 (Security of ΔRG [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 ΔRG)**.** *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,* $\Delta\mathsf{RG} \in (\mathbb{C},\ \deg d)$*, 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 $(\mathbb{C}, \text{deg } d)$ 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.

- $\Delta\mathsf{RG.SetupPoly}(1^\lambda, 1^n, 1^B)$: Run $\mathsf{sPRG.IdSamp}(1^\lambda, 1^n) \to I'$, and output $I = (I', B, \lambda, n)$.
- $\Delta\mathsf{RG.SetupSeed}(I)$: Run $\mathsf{sPRG.SdSamp}(I') \to (P, S)$ and output $\mathsf{sd} = (P, S)$.
- $\Delta\mathsf{RG.Eval}(I, \mathsf{sd})$: Compute $\boldsymbol{z} \leftarrow \mathsf{sPRG.Eval}(I', \mathsf{sd})$ 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 $m \geq m't$, $\Delta\mathsf{RG.Eval}(I, \mathsf{sd})$ outputs $\boldsymbol{y}$ where $y_i = \sum_{j\in[t]} 2^{j-1} \cdot z_{(i-1)\cdot t+j}$, and $\boldsymbol{z} = \mathsf{sPRG.Eval}(I', \mathsf{sd})$. 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 $\Delta\mathsf{RG}$ 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 $\Delta\mathsf{RG}$ is non-zero and formed as above. Thus, this hybrid corresponds to the first challenge distribution in the security definition of $\Delta\mathsf{RG}$ (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 $\Delta\mathsf{RG}$ 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. □

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* PPGen, Setup, 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 $\boldsymbol{A}, \boldsymbol{b} = \boldsymbol{s}\boldsymbol{A} + 2\boldsymbol{e} + \boldsymbol{x}$ 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

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

  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 $f(\boldsymbol{x}) + 2\boldsymbol{e}'$

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

  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 $(\mathsf{arith\text{-}NC}^1, \deg 2)$.

$\mathsf{Dec}(\mathsf{sk}, \mathsf{ct})$**:** 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 $k^{\lceil \frac{D}{2} \rceil} \log p = O(\lambda^{(\lceil \frac{D}{2} \rceil / \epsilon) + 1}) = \mathsf{poly}(\lambda)$.

Factoring and Pairings are not Necessary for iO: Circular-Secure LWE Suffices

Zvika Brakerski*[1], Nico Döttling[2], Sanjam Garg†[3], and Giulio Malavolta[4]

[1]Weizmann Institute of Science
[2]CISPA Helmoltz Center for Information Security
[3]UC Berkeley
[4]Max Planck Institute for Security and Privacy

Abstract

We construct indistinguishability obfuscation (iO) solely under circular-security properties of encryption schemes based on the Learning with Errors (LWE) problem. Circular-security assumptions were used before to construct (non-leveled) fully-homomorphic encryption (FHE), but our assumption is stronger and requires circular randomness-leakage-resilience. In contrast with prior works, this assumption can be conjectured to be post-quantum secure; yielding the first provably secure iO construction that is (plausibly) post-quantum secure.

Our work follows the high-level outline of the recent work of Gay and Pass [ePrint 2020], who showed a way to remove the heuristic step from the homomorphic-encryption based iO approach of Brakerski, Döttling, Garg, and Malavolta [EUROCRYPT 2020]. They thus obtain a construction proved secure under circular security assumption of natural homomorphic encryption schemes — specifically, they use homomorphic encryption schemes based on LWE and DCR, respectively. In this work we show how to remove the DCR assumption and remain with a scheme based on the circular security of LWE alone. Along the way we relax some of the requirements in the Gay-Pass blueprint and thus obtain a scheme that is secure under a relaxed assumption. Specifically, we do not require security in the presence of a key-cycle, but rather only in the presence of a key-randomness cycle.

1 Introduction

The goal of program obfuscation [Had00, BGI+01] is to transform an arbitrary circuit Π into an unintelligible but functionally equivalent circuit $\tilde{\Pi}$. The aforementioned works showed that strong simulation-based notions of obfuscation were impossible for general purpose functionalities. However, the seemingly weaker *indistinguishability obfuscation* (iO) was not ruled out by prior work (and has in fact been shown to be the same as the best possible notion of obfuscation [GR07]). In broad terms, iO requires that if two circuits Π_0 and Π_1 are two implementations of the same function, then their obfuscations are computationally indistinguishable.

Garg et al. [GGH13a, GGH+13b] presented the first candidate for general purpose iO, paving the way for numerous other candidates based on a variety of mathematical structures. Although iO appears to be a weak notion of security, it has been shown to be sufficient for numerous cryptographic applications,

*Supported by the Binational Science Foundation (Grant No. 2016726), and by the European Union Horizon 2020 Research and Innovation Program via ERC Project REACT (Grant 756482) and via Project PROMETHEUS (Grant 780701).

†Supported in part from AFOSR Award FA9550-19-1-0200, AFOSR YIP Award, NSF CNS Award 1936826, DARPA and SPAWAR under contract N66001-15-C-4065, DARPA/ARL SAFEWARE Award W911NF15C0210, a Hellman Award and research grants by the Okawa Foundation, Visa Inc., and Center for Long-Term Cybersecurity (CLTC, UC Berkeley). The views expressed are those of the authors and do not reflect the official policy or position of the funding agencies.

including ones that were previously not known to exist under other assumptions (see [SW14, GGHR14, BZ14] for examples). The first realizations of obfuscation relied an a new algebraic object called multilinear maps [GGH13a, CLT13, GGH15], which had only recently been constructed. Furthermore, the security of these objects relied on new (and poorly understood) computational intractability assumptions, or more commonly on plain heuristics. In fact, several attacks on multilinear map candidates [CHL$^+$15, HJ16] and on obfuscation constructions based on multilinear maps [MSZ16, CGH17] were demonstrated. To defend against these attacks, several safeguards have been (e.g., [GMM$^+$16, CVW18, MZ18, BGMZ18, DGG$^+$18]) proposed. Even with these heuristic safeguards, all but the schemes based on the Gentry et al. [GGH15] multilinear maps are known to broken against quantum adversaries.

Towards the goal of avoiding heuristics and obtaining provably secure constructions, substantial effort was made towards obtaining iO while minimizing (with the ultimate goal of removing) the use of multilinear maps [Lin16, LV16, AS17, Lin17, LT17]. These efforts culminated in replacing the use of multilinear maps with just bilinear maps [Agr19, JLMS19, AJL$^+$19], together with an additional pseudorandom generator with special properties. Very recently this last limitation was removed by Jain, Lin and Sahai [JLS20]. Specifically, they obtained iO based on the combined (sub-exponential) hardness of the Learning with Errors problem (LWE), a large-modulus variant of the Learning Parity with Noise problem (LPN), the existence of a pseudorandom generator in NC_0, and in addition the hardness of the external Diffie-Hellman problem in bilinear groups (SXDH). We note that the use of the pairings makes these construction insecure against quantum adversaries.

A different approach towards provably secure iO, which is more relevant to this work, was presented by Brakerski et al. [BDGM20]. They showed an iO candidate that is based on combining certain natural *homomorphic* encryption schemes. However, their construction was *heuristic* in the sense that security argument could only be presented in the random oracle model. In a recent work, Gay and Pass [GP20] showed a way to remove the heuristic step and instead rely on a concrete assumption. Their construction is proved secure under the circular security of natural homomorphic encryption schemes — specifically, they use homomorphic encryption schemes based on LWE and Decisional Composite Residuosity (DCR, also known as Paillier's assumption). In terms of assumptions, their construction assumes sub-exponential security of (i) the Learning with Error (LWE) assumption, (ii) the Decisional Composite Residuosity (DCR) assumption, and (iii) a new notion of security that they call "shielded randomness leakage" (SRL). The latter essentially requires that a fully homomorphic encryption scheme (specifically the GSW encryption scheme [GSW13]) remains secure even in the presence of a key-cycle with the Damgård-Jurik encryption scheme [DJ01]. Moreover, the notion of security is not the standard semantic security, but rather a new notion of security with respect to leakage of ciphertext randomness. We note that this construction is insecure against quantum attackers because of the use of the Damgård-Jurik encryption scheme [DJ01].[1] In this work, we ask:

Can we realize provably secure constructions of iO based solely on hard problems in lattices?

1.1 Our results

We obtain a general purpose iO construction based solely on the circular security of LWE-based encryption schemes. This is done by presenting a "packed" variant of the dual-Regev LWE-based encryption scheme, and showing novel ways of manipulating ciphertexts of this variant in conjunction with ciphertexts of an FHE scheme. This allows us to remove the need for DCR-based encryption from the construction of [BDGM20, GP20]. Furthermore, our technique allows to relax the SRL security property that is required, so that we no longer need to require SRL security with respect to a key-cycle, but rather only with respect to a key-randomness cycle. We put forth this potentially weaker assumption as an object for further study.

The security of our construction relies thus on a circular-security assumption (either SRL security as in [GP20] or our weaker key-randomness circularity notion). The term "circular security" refers to the notion where the security properties of an encryption scheme are preserved even when given an encryption of the secret key itself, or, as in this case, given a "key cycle" where the key of the first scheme is encrypted

[1] Later, [GP20] updated their manuscript to also include a solution based on LWE. See Section 1.3 for additional discussion.

using the second scheme, and the key of the second scheme is encrypted using the first scheme. Circular security assumptions are commonly used in the literature. Notably, they are known to imply such primitives as "unrestricted" (non-leveled) fully homomorphic encryption (FHE) schemes via Gentry's bootstrapping paradigm [Gen09].

Concretely, the circular assumption made in [GP20], and thus also in this work, is that a scheme (in particular a leveled FHE scheme) which enjoys the property that security is maintained even given some particular kind of leakage on the randomness of the ciphertext. Indeed, standard GSW encryption [GSW13] satisfies SRL security (under the LWE assumption), and the assumption we make is that SLR security holds even when given a key-cycle connecting GSW to another encryption scheme. The second scheme in [GP20] was based on DCR, whereas in our case this second encryption scheme is also based on LWE. While this assumption falls into the category of "circular security assumptions", similarly to the ones that underlie bootstrapping in FHE, the concrete assumption is quite different. While in the FHE setting it was only assumed that (standard) CPA security is preserved given a key cycle, here we assume that the stronger SRL property remains intact.

Let us now state our results somewhat more precisely.

Theorem 1.1 (Informal) *Assume the (sub-exponential) hardness of the LWE problem, and the SRL security of GSW in the presence of a 2-key cycle with a packed variant of dual-Regev, then there exists indistinguishability obfuscation for all circuits.*

We note that if we further assume that circular security also maintains post-quantum security, then our assumption becomes post-quantum secure; yielding the first provably secure iO construction that is post-quantum secure. Additionally, our techniques yield also a scheme provably secure against the (potentially) weaker assumption of key-randomness SRL security, i.e. that SRL security is retained in the presence of a (packed) dual Regev encryption of the GSW secret key and a GSW encryption of the randomness used in such a ciphertext.

Theorem 1.2 (Informal) *Assume the (sub-exponential) hardness of the LWE problem, and the SRL security of GSW in the presence of a randomness-key cycle with a packed variant of dual-Regev, then there exists indistinguishability obfuscation for all circuits.*

1.2 Technical Overview

We now provide a technical outline of our construction and its properties.

Obfuscation via Homomorphic Encryption. The connection between (fully) homomorphic encryption and obfuscation is fairly straightforward. Given a program Π to be obfuscated, we can provide a ciphertext c_Π which encrypts Π under an FHE scheme. This will allow to use homomorphism to derive $c_x = \mathsf{Enc}(\Pi(x))$ for all x. Now all that is needed is a way to decrypt c_x in a way that does not reveal any information on Π. Early works (e.g. [GGH$^+$13b] and followups) attempted to use this approach and provide a "defective" version of the secret key of the FHE scheme, but a different approach was suggested in [BDGM20].

Specifically, [BDGM20] considered a homomorphic evaluation that takes c_Π to c_{TT}, an encryption of the *entire truth table* of Π, i.e. to an encryption of a multi-bit value. By relying on prior generic transformations [LPST16], they showed that one can reduce the task of constructing general-purpose obfuscation to the task of computing a "decryption" hint for c_{TT} with the following properties:

- Succinctness: The size of the decryption hint must be sublinear in the size of the truth table $|\mathsf{TT}|$.
- Simulatability: The decryption hint should not reveal any additional information besides the truth table TT.

The reason why this is helpful is that some so-called "packed-encryption" schemes have the property that a short ciphertext-dependent decryption hint suffices in order to decrypt the ciphertext, in a way that does

not seem to leak the secret key of the scheme itself. While standard FHE schemes do not natively support packed encryption, it was shown in [BDGM19] that it is possible to use the so-called key-switching technique to switch from an FHE scheme into a packed-encryption scheme.

Alas, when instantiating the components of the [BDGM20] approach in its simplistic form described above, the decryption hint leaks information that renders the scheme insecure. To counter this issue, [BDGM20] proposed to inject another source of randomness: By adding freshly sampled ciphertexts of the packed-encryption scheme (which in their case was instantiated with the Damgård-Jurik scheme [DJ01]) one can smudge the leakage of the decryption hint. However the size of these fresh ciphertext would largely exceed the size of the truth table TT. Therefore, [BDGM20] proposed to heuristically sample them from a random oracle, leveraging the fact that the ciphertexts of [DJ01] are *dense*, i.e. a uniformly sampled string lies in the support of the encryption algorithm with all but negligible probability. This led to a candidate, but without a proof of security.

A Provably Secure Scheme. In a recent work, Gay and Pass [GP20] observed that for the purpose of constructing obfuscation, it suffices to consider schemes in the common random string (CRS) model where, importantly, the size of the CRS can exceed the size of the truth table. This allowed them to place the Damgård-Jurik ciphertexts in the CRS and therefore avoid relying on random-oracle-like heuristics.

They propose a new method to prove the security of this approach: Leveraging the structural property of the GSW scheme [GSW13]. They showed that adding a GSW encryption of 0 to the evaluated FHE ciphertext (before key-switching to Damgård-Jurik) allows one to program the FHE ciphertext in the security proof. To sample these GSW encryptions of 0, they propose to draw the random coins $\mathbf{r}^*$ again from the CRS and let the evaluator recompute the correct ciphertext $\mathsf{GSW.Enc}(0; \mathbf{r}^*)$.

Taken together, these new ideas allow them to prove their construction secure against the shielded randomness leakage (SRL) security of the resulting FHE scheme. Loosely speaking, SRL security requires that semantic security of an encryption scheme is retained in the presence of an oracle that leaks the randomness $\mathbf{r}_f$ of the homomorphic evaluation of the function f over the challenge ciphertext. However the randomness $\mathbf{r}_f$ is not revealed in plain to the adversary, instead it is "shielded" by the random coins of a fresh GSW ciphertext $c = \mathsf{GSW.Enc}(0; \mathbf{r}^*)$. That is, the adversary is given $(\mathbf{r}_f - \mathbf{r}^*, c)$. In fact, the adversary can obtain polynomially-many samples from this distribution, for any function f, conditioned on the fact that the adversary knows the output of $f(m^*)$, where m^* is the hidden message.

To gain confidence in the veracity of the assumption, [GP20] show that the GSW encryption scheme satisfies SRL security if the (plain) LWE assumption holds. However, their obfuscation scheme requires one to publish a key cycle of GSW and Damgård-Jurik (i.e. an encryption of the GSW secrey key under Damgård-Jurik and vice versa). Thus their final assumption is that SRL security is retained in the presence of such a key cycle.

Obfuscation from Circular-Secure LWE. We wish to remove the need for the Damgård-Jurik encryption scheme from the above construction paradigm. The major obstacle to overcome consists is designing an LWE-based encryption scheme that simultaneously satisfies three properties.

- Linear Homomorphism: In order to key switch the GSW ciphertext into this form, the scheme must satisfy some weak notion of homomorphism. Specifically, it must support the homomorphic evaluation of linear functions.
- Succinct Randomness: The scheme must allow us to encrypt a long message string with a short randomness, that can then function as the decryption hint.
- Dense Ciphertexts: A uniformly sampled string must lie in the support of the encryption algorithm with all but negligible probability. This will allow us to parse the CRS as a collection of ciphertexts.[2]

[2]Note that for the purpose of constructing the obfuscator, one could make do with a common reference string which can have an arbitrary distribution. However, the string needs to be parsed as a ciphertext with respect to *all* public-keys. Requiring dense ciphertexts is a simple requirement that implies this property.

Unfortunately all natural lattice-based candidates seem to fail to satisfy all of these properties. In particular, for all LWE-based schemes linear homomorphism seems to be at odds with dense ciphertexts: To ensure that the noise accumulated during the homomorphic evaluation does not impact the decryption correctness, one needs to ensure a gap between the noise bound and the modulus. More concretely, ciphertext are typically of the form $(\mathbf{a}, \mathbf{a} \cdot \mathbf{s} + e + q/2 \cdot m) \in \mathbb{Z}_q^{n+1}$ where $e \ll q$, which makes them inherently sparse.

Our Solution: A Packed Variant of Dual-Regev that is also Dense-Friendly. We show that the above requirements can be relaxed. Our starting point is devising a "packed" version of the dual-Regev encryption scheme [GPV08]. This scheme will not have dense ciphertexts so it does not fit the requirements from previous works. However, we will show how we can define, for the same scheme, a family of ciphertexts which are both "almost dense" and can inter-operate with the non-dense scheme, so as to allow to construct the obfuscator.

Let us start with our packed dual-Regev scheme. To pack a k-bit plaintext $\mathbf{m} \in \{0,1\}^k$ in a dual-Regev ciphertext we construct the public key as a matrix $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$, which is statistically close to uniform but is sampled together with a trapdoor τ (whose role will be explained below), and another uniformly sampled matrix $\mathbf{B} \in \mathbb{Z}_q^{k \times n}$. The encryption algorithm computes a the ciphertext as

$$(\mathbf{A} \cdot \mathbf{r} + \mathbf{e}_0, \mathbf{B} \cdot \mathbf{r} + q/2 \cdot \mathbf{m} + \mathbf{e})$$

where $\mathbf{r} \leftarrow_\$ \mathbb{Z}_q^n$ is the encryption randomness and the vectors $\mathbf{e}_0$ and $\mathbf{e}$ are the encryption noises, where the norm of both vectors is bounded by some $B \ll q$. The property of the trapdoor τ is that it allows to recover $\mathbf{r}$ from $\mathbf{A} \cdot \mathbf{r} + \mathbf{e}_0$. The (semantic) security of the scheme follows directly by definition of LWE. To decrypt, therefore, one can first use the trapdoor τ to recover $\mathbf{r}$ from the first m elements of the ciphertext, and then recompute the mask $\mathbf{B} \cdot \mathbf{r}$ and recover each individual bit by rounding to the closest multiple of $q/2$. Setting the parameters appropriately, we can guarantee that the decryption is always successful. One important property of this scheme is that the random coins $\mathbf{r} \in \mathbb{Z}_q^n$ are sufficient to recover the entire message and furthermore the size of $\mathbf{r}$ is succinct (in particular independent of k).

In terms of homomorphism, the scheme is straightforwardly additively homomorphic. Furthermore, it supports key switching from any scheme with almost-linear decryption as per [BDGM19].[3] In particular it is possible to take a (long) message encrypted under an FHE scheme such as GSW and convert it to an encryption of the same message under packed dual-Regev, using precomputed key-switching parameters.[4]

As explained above, this scheme does not have dense ciphertexts. At this point we make two crucial observations that will allow us to bypass this hurdle.

(1) In order to construct the obfuscator using the [BDGM20] approach, dense ciphertexts only need to enjoy a very limited form of homomorphism, they only need to support a single addition with a non-dense ciphertext.

This is essentially because the obfuscator has the following outline. It starts by considering the dense ciphertext from the CRS (or oracle in the case of the original [BDGM20]), and homomorphically bootstraps it into a non-dense FHE ciphertext by evaluating the decryption circuit. Let $\mathbf{m}$ be the (random) message that is induced by the process. Then, the FHE encryption of $\mathbf{m}$ is processed in order to create a non-dense *packed* encryption of $\mathbf{m} \oplus \mathsf{TT}$, where TT is the truth table of the program to be obfuscated (or, more accurately, a chunk of this truth table, partitioning into chunks is required in order to allow reusability of the keys). Then a single homomorphic addition between the dense and non-dense ciphertext would imply a packed encryption of the truth table. All of this can be performed by the evaluator of the obfuscated program, so all that is needed is the decryption hint for this final ciphertext, that would allow to recover TT.

We note importantly, that in prior approaches (including the [GP20] blueprint) the aforementioned bootstrapping creates a key cycle, since a packed ciphertext is bootstrapped into an FHE ciphertext, which is afterwards key-switched into a packed ciphertext. However, we notice that it suffices to provide an

[3] This is done using the by-now-standard technique of encrypting powers-of-two of the elements of the secret key of the latter scheme, so that it is possible to evaluate any inner product homomorphically.

[4] We not that the key switching parameters are quite long so it is required for our method that they are reusable.

encryption of the (succinct) *randomness* of the dense ciphertext in order to apply bootstrapping, thus leading to a relaxed key-randomness circular assumption. Interestingly, this observation is not very useful for actual dense ciphertexts (since finding the randomness would require using the key), however, our relaxed notion of density described below will allow to apply it and thus relax the circularity notion as well.

(2) A notion of *almost-everywhere* density suffices. A ciphertext distribution is almost-everywhere dense if it is dense except for a non-dense part whose length is independent of k (the message length).

The reason that this is sufficient is that the non-dense part of the ciphertext, which we refer to as the *header*, can be generated by the obfuscator and provided to the evaluator as a part of the obfuscated program. Since the header is short, and in particular the message length k can be selected to be much longer than the header, the effect on the length of the obfuscated program will be minimal.

As hinted above, since the obfuscator generates the header, it in particular also samples the randomness for the final almost-everywhere dense ciphertext. This means that the obfuscator can generate the bootstrapping parameters using this randomness without requiring a key cycle.

Dense Encryption Mode. With these observations in mind we describe an alternative encryption mode ($\mathsf{DenseEnc}$) for the packed variant of dual-Regev where the bulk of the ciphertext is dense. On input a message $\mathbf{m} \in \{0,1\}^k$, the encryption algorithm in dense mode computes the following ciphertext

$$(\mathbf{A} \cdot \mathbf{r} + \mathbf{e}_0, \mathbf{B} \cdot \mathbf{r} + q/2 \cdot \mathbf{m} + \mathbf{u})$$

where $\mathbf{r}$ and $\mathbf{e}_0$ are sampled as before and $\mathbf{u} \leftarrow\!\!\$\, [-q/4, +q/4]^k$. For convenience, we are going to split the ciphertexts into two blocks: The header $\mathbf{h}_0 \in \mathbb{Z}_q^m$ and the message carrier $(h_1, \ldots, h_k) \in \mathbb{Z}_q^k$. Foremost, observe that the decryption algorithm as described before still returns the correct message with probability 1, since it recovers the same $\mathbf{r}$ from $\mathbf{h}_0$. Furthermore, note that (for a fixed header) all vectors $(h_1, \ldots, h_k) \in \mathbb{Z}_q^k$ are in the support of the encryption algorithm. Since $k \gg m$, most of the elements of the ciphertext in the alternative decryption mode are dense.

One can verify that the aforementioned limited form of homomorphism indeed holds, namely that

$$\mathsf{dR.Enc}(\mathbf{m}) + \mathsf{dR.DenseEnc}(\mathbf{m}') \in \mathsf{dR.DenseEnc}(\mathbf{m}_0 \oplus \mathbf{m}').$$

This is the case since

$$\begin{aligned}
&(\mathbf{A} \cdot \mathbf{r} + \mathbf{e}_0, \mathbf{B} \cdot \mathbf{r} + q/2 \cdot \mathbf{m} + \mathbf{e}) + (\mathbf{A} \cdot \mathbf{r}' + \mathbf{e}_0', \mathbf{B} \cdot \mathbf{r}' + q/2 \cdot \mathbf{m}' + \mathbf{u}) \\
&= (\mathbf{A} \cdot (\mathbf{r} + \mathbf{r}') + \mathbf{e}_0 + \mathbf{e}_0', \mathbf{B} \cdot (\mathbf{r} + \mathbf{r}') + q/2 \cdot (\mathbf{m} \oplus \mathbf{m}') + \mathbf{e} + \mathbf{u}) \\
&= (\mathbf{A} \cdot \tilde{\mathbf{r}} + \tilde{\mathbf{e}}_0, \mathbf{B} \cdot \tilde{\mathbf{r}} + q/2 \cdot (\mathbf{m} \oplus \mathbf{m}') + \tilde{\mathbf{u}})
\end{aligned}$$

where $\tilde{\mathbf{u}} = \mathbf{e} + \mathbf{u} \in [-q/4, +q/4]^k$ with all but negligible probability over the random choice of $\mathbf{u}$, for an appropriate choice of the parameters.

Doing Away with the Header. We notice that given our two observations above, the goal of the header in the obfuscation scheme is quite minimal. The header is not needed for homomorphism, and is only needed for the purpose of extracting the randomness $\mathbf{r}$ at decryption time. We then observe that decrypting packed ciphertext is done in two contexts in the scheme. The first is when we bootstrap the almost-everywhere dense ciphertext into an FHE ciphertext, and the other is when the evaluator of the obfuscated program recovers TT from the final ciphertext. For the latter there is no need for a header since the decryption hint, i.e. the respective $\mathbf{r}$ value, is provided within the obfuscated program. For the former we do not need a header of a specific structure, but rather simply an encryption of $\mathbf{r}$ that allows bootstrapping the almost-dense ciphertext. It therefore suffices to provide $\mathsf{GSW.Enc}(\mathbf{r})$ directly, which makes the header completely redundant.

On the Assumption. Equipped with the newly developed packed version of dual-Regev we can follow the [BDGM20, GP20] approach, with the aforementioned modifications, to construct the obfuscator. The resulting construction can be shown secure against the assumption that the SRL security of GSW is retained in the presence of a key cycle with the packed dual-Regev encryption scheme as presented above.

We then observe that it suffices to assume SRL security with respect to key-randomness cycles, rather than key cycles. We note that this assumption is no-stronger than key-cycle SRL since given a key-cycle it is possible to homomorphically generate a key-randomness cycle, but the converse is not known to be true.

Adding this to our observation about the redundancy of the header, the assumption we require is that SRL security is retained in the presence of a key-randomness cycle between GSW and packed dual-Regev, i.e.

$$(\mathsf{GSW.Enc}(\mathbf{r}), \mathsf{dR.Enc}(\mathsf{sk}_{\mathsf{GSW}}; \mathbf{r}))\,.$$

Since dual-Regev is randomness recoverable, this assumption is (potentially) strictly weaker than SRL security in the presence of a key-cycle.

1.3 Other Related Work

Subsequently to the posting of this manuscript online (but concurrently and independently) [GP20] updated their manuscript to include a solution based on LWE in the place of DCR. They do not make the observations that a relaxed notion of density suffices (and is preferable) and thus they explicitly construct an encryption scheme with dense ciphertexts based on the (primal) Regev encryption scheme. The resulting scheme is more involved and in particular requires the circular SRL security of GSW rather than the relaxed key-randomness circularity notion.

Wee and Wichs [WW20], again concurrently, presented another instantiation of the [BDGM20] approach which is arguably post-quantum secure. They rely on an indistinguishability assumption between two distributions and not directly on circular security. However, the underlying machinery developed shares many similarities with our approach. Specifically, while we essentially rely on randomness that is embedded in the CRS by interpreting it as an obliviously sampled ciphertext (which thus corresponds to one encrypted with fresh randomness), their approach is to use a pseudorandom function to transform the CRS into a randomizer for the output hint.

2 Preliminaries

We denote by $\lambda \in \mathbb{N}$ the security parameter. We say that a function negl is negligible if it vanishes faster than any polynomial. Given a set S, we denote by $s \leftarrow\!\!\$\, S$ the uniform sampling from S. We say that an algorithm is PPT if it can be implemented by a probabilistic machine running in time $\mathsf{poly}(\lambda)$. We say that two distributions (D_0, D_1) are computationally (statistically, resp.) indistinguishable if for all PPT (unbounded, resp.) distinguishers, the probability to tell D_0 an D_1 apart is negligibly close to $1/2$. Matrices are denoted by $\mathbf{M}$ and vectors are denoted by $\mathbf{v}$. We denote the infinity norm of a vector $\mathbf{v}$ by $\|\mathbf{v}\|_\infty$. We recall the smudging lemma [AIK11, AJL+12].

Lemma 2.1 (Smudging) *Let $B_1 = B_1(\lambda)$ and $B_2 = B_2(\lambda)$ be positive integers and let $e_1 \in [-B_1, B_1]$ be a fixed integer. Let $e_2 \leftarrow\!\!\$\, [-B_2, B_2]$ chosen uniformly at random. Then the distribution of e_2 is statistically indistinguishable to that of $e_2 + e_1$ as long as $B_1/B_2 = \mathsf{negl}(\lambda)$.*

2.1 Indistinguishability Obfuscation

We recall the notion of indistinguishability obfuscation (iO) from [GGH+13b].

Definition 2.2 (Indistinguishability Obfuscation) *A PPT machine iO is an indistinguishability obfuscator for a circuit class $\{\mathfrak{C}_\lambda\}_{\lambda\in\mathbb{N}}$ if the following conditions are satisfied:*

(Functionality) For all $\lambda \in \mathbb{N}$, all circuit $\Pi \in \mathfrak{C}_\lambda$, all inputs x it holds that: $\tilde{\Pi}(x) = \Pi(x)$, where $\tilde{\Pi} \leftarrow \mathsf{iO}(\Pi)$.

(Indistinguishability) For all $\lambda \in \mathbb{N}$, all pairs of circuit $(\Pi_0, \Pi_1) \in \mathfrak{C}_\lambda$ such that $|\Pi_0| = |\Pi_1|$ and $\Pi_0(x) = \Pi_1(x)$ on all inputs x, it holds that the following distributions are computationally indistinguishable:

$$\mathsf{iO}(\Pi_0) \approx \mathsf{iO}(\Pi_1).$$

XiO. We recall a theorem from Lin et al. [LPST16], that states that (assuming the hardness of the LWE problem), constructing an obfuscator for circuits with logarithmically-many input bits suffices to build general-purpose obfuscation.

Theorem 2.3 (XiO) *Assuming sub-exponentially hard LWE, if there exists a sub-exponentially secure indistinguishability obfuscator (with pre-processing) for $\mathsf{P}^{\mathsf{log}}/\mathsf{poly}$ with non-trivial efficiency, then there exists an indistinguishability obfuscator for P/poly with sub-exponential security.*

Here $\mathsf{P}^{\mathsf{log}}/\mathsf{poly}$ denotes the class of polynomial-size circuits with inputs of length $\eta = O(\log(\lambda))$ and by non-trivial efficiency we mean that the size of the obfuscated circuit is bounded by $\mathsf{poly}(\lambda, |\Pi|) \cdot 2^{\eta \cdot (1-\varepsilon)}$, for some constant $\varepsilon > 0$. Note that the above theorem poses no restriction on the runtime of the obfuscator. Furthermore, the theorem allows the obfuscator to access a large uniform random string (the pre-processing) of size even larger than the truth table of the circuit.

2.2 Learning with Errors

We recall the definition of the learning with errors (LWE) problem [Reg05].

Definition 2.4 (Learning with Errors) *The LWE problem is parametrized by a modulus q, positive integers (n, m) and an error distribution χ. The LWE problem is hard if the following distributions are computationally indistinguishable:*

$$(\mathbf{A}, \mathbf{A} \cdot \mathbf{s} + \mathbf{e}) \approx (\mathbf{A}, \mathbf{u})$$

where $\mathbf{A} \leftarrow_\$ \mathbb{Z}_q^{m \times n}$, $\mathbf{s} \leftarrow_\$ \mathbb{Z}_q^n$, $\mathbf{u} \leftarrow_\$ \mathbb{Z}_q^m$, and $\mathbf{e} \leftarrow_\$ \chi^m$.

As shown in [Reg05, PRS17], for *any* sufficiently large modulus q the LWE problem where χ is a discrete Gaussian distribution with parameter $\sigma = \alpha q \geq 2\sqrt{n}$ (i.e. the distribution over $\mathbb{Z}$ where the probability of x is proportional to $e^{-\pi(|x|/\sigma)^2}$), is at least as hard as approximating the shortest independent vector problem (SIVP) to within a factor of $\gamma = \tilde{O}(n/\alpha)$ in *worst case* dimension n lattices. We refer to $\alpha = \sigma/q$ as the *modulus-to-noise* ratio, and by the above this quantity controls the hardness of the LWE instantiation. Hereby, LWE with polynomial α is (presumably) harder than LWE with super-polynomial or sub-exponential α. We can truncate the discrete Gaussian distribution χ to $\sigma \cdot \omega(\sqrt{\log(\lambda)})$ while only introducing a negligible error. Consequently, we typically omit the actual distribution χ but only use the fact that it can be bounded by a (small) value B.

2.3 Public-Key Encryption

We recall the definition of public key encryption in the following.

Definition 2.5 (Public-Key Encryption) *A public-key encryption scheme consists of the following efficient algorithms.*

$\underline{\mathsf{KeyGen}(1^\lambda)}$**:** *On input the security parameter 1^λ, the key generation algorithm returns a key pair $(\mathsf{sk}, \mathsf{pk})$.*

$\underline{\mathsf{Enc}(\mathsf{pk}, m)}$**:** *On input a public key pk and a message m, the encryption algorithm returns a ciphertext c.*

$\underline{\mathsf{Dec}(\mathsf{sk}, c)}$**:** *On input the secret key sk and a ciphertext c, the decryption algorithm returns a message m.*

Correctness and Semantic Security. We recall the standard notions of correctness and semantic security [GM82] for public-key encryption.

Definition 2.6 (Correctness) *A public-key encryption scheme* $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *is correct if for all* $\lambda \in \mathbb{N}$*, all messages* m*, all* $(\mathsf{sk}, \mathsf{pk})$ *in the support of* $\mathsf{KeyGen}(1^\lambda)$*, and all* c *in the support of* $\mathsf{Enc}(\mathsf{pk}, m)$ *it holds that* $\mathsf{Dec}(\mathsf{sk}, c) = m$*.*

Definition 2.7 (Semantic Security) *A public-key encryption scheme* $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *is semantically secure if for all* $\lambda \in \mathbb{N}$*, all pairs of message* (m_0, m_1)*, it holds that the following distributions are computationally indistinguishable*

$$(\mathsf{pk}, \mathsf{Enc}(\mathsf{pk}, m_0)) \approx (\mathsf{pk}, \mathsf{Enc}(\mathsf{pk}, m_1))$$

where $(\mathsf{sk}, \mathsf{pk}) \leftarrow_\$ \mathsf{KeyGen}(1^\lambda)$*.*

Homomorphic Encryption. We say that a public-key encryption scheme $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ is homomorphic for the circuit class $\{\mathfrak{C}_\lambda\}_{\lambda\in\mathbb{N}}$ if there exist an efficient deterministic algorithm Eval such that for all $\Pi \in \mathfrak{C}_\lambda$, all $(\mathsf{sk}, \mathsf{pk})$ in the support of KeyGen, all vectors of messages $(m_1, \ldots, m_\mu)$, all ciphertexts $(c_1, \ldots, c_\mu)$ in the support of $(\mathsf{Enc}(\mathsf{pk}, m_1), \ldots, \mathsf{Enc}(\mathsf{pk}, m_\mu))$ it holds that

$$\mathsf{Dec}(\mathsf{sk}, \mathsf{Eval}(\mathsf{pk}, \Pi, (c_1, \ldots, c_\mu))) = \Pi(m_1, \ldots, m_\mu).$$

Furthermore, we say that a scheme is *fully-homomorphic* if it is homomorphic for all polynomial-size circuits.

Circular Security. We say that two encryption schemes $(\mathsf{KeyGen}_0, \mathsf{Enc}_0, \mathsf{Dec}_0)$ and $(\mathsf{KeyGen}_1, \mathsf{Enc}_1, \mathsf{Dec}_1)$ form a key cycle if the distinguisher is given a cross-encryption of the secret keys, i.e. $\mathsf{Enc}(\mathsf{pk}_1, \mathsf{sk}_0)$ and $\mathsf{Enc}(\mathsf{pk}_0, \mathsf{sk}_1)$. We say that the scheme is 2-circular secure if semantic security is retained in the presence of such a cycle.

Definition 2.8 (2-Circular Security) *A pair of public-key encryption schemes* $(\mathsf{KeyGen}_0, \mathsf{Enc}_0, \mathsf{Dec}_0)$ *and* $(\mathsf{KeyGen}_1, \mathsf{Enc}_1, \mathsf{Dec}_1)$ *is 2-circular secure if for all* $\lambda \in \mathbb{N}$*, all pairs of message* (m_0, m_1)*, it holds that the following distributions are computationally indistinguishable*

$$\begin{aligned}&(\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{Enc}_0(\mathsf{pk}_0, \mathsf{sk}_1), \mathsf{Enc}_1(\mathsf{pk}_1, \mathsf{sk}_0), \mathsf{Enc}_0(\mathsf{pk}_0, m_0))\\ &\approx (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{Enc}_0(\mathsf{pk}_0, \mathsf{sk}_1), \mathsf{Enc}_1(\mathsf{pk}_1, \mathsf{sk}_0), \mathsf{Enc}_0(\mathsf{pk}_0, m_1))\end{aligned}$$

where $(\mathsf{sk}_0, \mathsf{pk}_0) \leftarrow_\$ \mathsf{KeyGen}_0(1^\lambda)$ *and* $(\mathsf{sk}_1, \mathsf{pk}_1) \leftarrow_\$ \mathsf{KeyGen}_1(1^\lambda)$*.*

2.4 The GSW Fully-Homomorphic Encryption

In the following we briefly recall the encryption scheme by Gentry, Sahai, and Waters [GSW13] (henceforth, GSW). We denote by $n = n(\lambda)$ the lattice dimension and by $q = q(\lambda)$ the modulus (which we assume for simplicity to be even). We set $m > n\log(q)$ and $d = d(\lambda)$ as a bound on the depth of the arithmetic circuit to be evaluated.

$\underline{\mathsf{KeyGen}(1^\lambda)}$**:** Sample a uniform matrix $\mathbf{A} \leftarrow_\$ \mathbb{Z}_q^{n\times m}$ and a vector $\mathbf{s} \leftarrow_\$ \chi^n$. Set the public key to $(\mathbf{A}, \mathbf{b} = \mathbf{s}^T\mathbf{A} + \mathbf{e}^T)$, where $\mathbf{e} \leftarrow_\$ \chi^m$. The secret key is set to $(-\mathbf{s}, 1)$.

$\underline{\mathsf{Enc}(\mathsf{pk}, m)}$**:** On input a message $m \in \{0,1\}$, sample a uniform $\mathbf{R} \leftarrow_\$ \{0,1\}^{m\times m}$ and compute

$$\mathbf{C} = (\mathbf{A}, \mathbf{b}) \cdot \mathbf{R} + m \cdot \mathbf{G}$$

where $\mathbf{G} = (1, 2, \ldots, 2^{\log(q)})^T \otimes I_{(n+1)}$ and $I_{(n+1)} \in \{0,1\}^{(n+1)\times(n+1)}$ denotes the identity matrix.

$\underline{\mathsf{Eval}(\mathsf{pk}, \Pi, (c_1, \ldots, c_\mu))}$**:** There exists a (deterministic) polynomial-time algorithm that allows one to compute any d-bounded depth arithmetic circuit $\Pi : \{0,1\}^n \to \{0,1\}$ homomorphically over a vector of ciphertexts $(c_1, \ldots, c_\mu)$. For details about this algorithm, we refer the reader to [GSW13]. For the purpose of this work, the only relevant information is that the evaluated ciphertext $\mathbf{c}_\Pi \in \mathbb{Z}_q^{(n+1)}$ is an $(n+1)$-dimensional vector. For multiple bits of output, the resulting ciphertext is defined to be the concatenation of the single-bit ciphertexts.

$\underline{\mathsf{Dec}(\mathsf{sk}, \mathbf{c})}$**:** We assue without loss of generality that the input ciphertext $\mathbf{c} \in \mathbb{Z}_q^{(n+1)}$ is the output of the evaluation algorithm. Such a ciphertext defines a linear function $\boldsymbol{\ell}_{\mathbf{c}}$ such that

$$\boldsymbol{\ell}_{\mathbf{c}}(\mathsf{sk}) = q/2 \cdot m + e$$

where $|e| \leq \hat{B} = (m+1)^d mB$. The message m is recovered by returning the most significant bit of the output.

Note that the decryption routine of GSW consists of the application of a linear function, followed by a rounding and we refer to this property as to *almost-linear* decryption. In a slight abuse of notation, we sometimes write $\mathsf{KeyGen}(1^\lambda; q)$ to denote the above key generation algorithm with a fixed modulus q.

Alternate Encryption. For convenience we also define a modified encryption algorithm, where the output ciphertexts consists of a single column vector. An additional difference is that we sample the randomness with norm $\tilde{B} = 2^\lambda \cdot \hat{B}$.

$\underline{\mathsf{ColEnc}(\mathsf{pk}, m)}$**:** On input a message m, sample a uniform $\mathbf{r} \leftarrow\!\!\$\, [-\tilde{B}, +\tilde{B}]^m$ and compute

$$\mathbf{c} = (\mathbf{A}, \mathbf{b}) \cdot \mathbf{r} + m.$$

The multi-bit version of such an algorithm is defined accordingly to output the concatenation of independently sampled ciphertexts. This algorithm is going instrumental for our scheme, although ciphertexts in this form no longer support the homomorphi evaluation of arbitrary circuits. We now recall a useful Lemma from [GP20].

Lemma 2.9 (GSW Smudging) *Let $\tilde{B} = 2^\lambda \cdot \hat{B}$. For all $\lambda \in \mathbb{N}$, for all $(\mathsf{sk}, \mathsf{pk})$ in the support of $\mathsf{KeyGen}(1^\lambda)$, for all messages $(m_1, \ldots, m_\mu)$, for all depth-d circuit $(\Pi_1, \ldots, \Pi_\tau)$, the following distributions are statistically indistinguishable*

$$\left(\begin{array}{l} (c_1, \ldots, c_\mu), (\mathbf{r}_1^*, \ldots, \mathbf{r}_\tau^*), \\ \mathsf{Eval}(\mathsf{pk}, \Pi_1, (c_1, \ldots, c_\mu)) + \mathsf{ColEnc}(\mathsf{pk}, 0; \mathbf{r}_1^*), \ldots, \mathsf{Eval}(\mathsf{pk}, \Pi_\tau, (c_1, \ldots, c_\mu)) + \mathsf{ColEnc}(\mathsf{pk}, 0; \mathbf{r}_\tau^*) \end{array}\right)$$
$$\approx \left(\begin{array}{l} (c_1, \ldots, c_\mu), (\mathbf{r}_1^* - \mathbf{r}_{\Pi,1}, \ldots, \mathbf{r}_\tau^* - \mathbf{r}_{\Pi,\tau}), \\ \mathsf{ColEnc}(\mathsf{pk}, \Pi_1(m_1, \ldots, m_\mu); \mathbf{r}_1^*), \ldots, \mathsf{ColEnc}(\mathsf{pk}, \Pi_\tau(m_1, \ldots, m_\mu); \mathbf{r}_\tau^*) \end{array}\right)$$

where $c_i \leftarrow\!\!\$\, \mathsf{Enc}(\mathsf{pk}, m_i)$, $\mathbf{r}_i^* \leftarrow\!\!\$\, [-\tilde{B}, +\tilde{B}]^m$, and $\mathbf{r}_{\Pi,i}$ is the randomness of the i-th evaluated ciphertext $\mathsf{Eval}(\mathsf{pk}, \Pi_i, (c_1, \ldots, c_\mu))$.

Shielded Randomness Leakage. The notion of shielded randomness leakage (SRL) security [GP20] says that the scheme is semantically secure even in the presence of an oracle that leaks some information about the randomness for evaluated ciphertext. The circuit to be evaluated homomorphically are fixed ahead of time (although they may depend on the challenge ciphertext) and the adversary is constrained to know the output of the evaluation ahead of time. We present a formal definition in the following.

Definition 2.10 (SRL Security) *A homomorphic encryption scheme $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec})$ is SRL secure if for all $\lambda \in \mathbb{N}$, all pairs of message (m_0, m_1), all $(\alpha_1, \ldots, \alpha_\tau)$, all circuits $(\Pi_1, \ldots, \Pi_\tau)$ such that*

for all $i = 1 \dots \tau$ it holds that $\Pi_i(m_0, \cdot) = \Pi_i(m_0, \cdot) = \alpha_\tau$, the following distributions are computationally indistinguishable

$$\begin{aligned}&\big(\mathsf{pk}, c = \mathsf{Enc}(\mathsf{pk}, m_0), \mathbf{c}_q^*, \dots, \mathbf{c}_\tau^*, \mathbf{r}_1^* - \mathbf{r}_1, \dots, \mathbf{r}_\tau^* - \mathbf{r}_\tau\big)\\ \approx\ &\big(\mathsf{pk}, c = \mathsf{Enc}(\mathsf{pk}, m_1), \mathbf{c}_q^*, \dots, \mathbf{c}_\tau^*, \mathbf{r}_1^* - \mathbf{r}_1, \dots, \mathbf{r}_\tau^* - \mathbf{r}_\tau\big)\end{aligned}$$

where $(\mathsf{sk}, \mathsf{pk}) \leftarrow\!\!\$\ \mathsf{KeyGen}(1^\lambda)$, $\mathbf{r}_i^* \leftarrow\!\!\$\ [-\tilde{B}, +\tilde{B}]$, $\mathbf{c}_i^* = \mathsf{ColEnc}(0; \mathbf{r}_i^*)$, and $\mathbf{r}_i$ is the randomness of the i-th evaluated ciphertext $\mathsf{Eval}(\mathsf{pk}, \Pi_i(\cdot, \mathbf{c}_i^*), c)$.

In [GP20], the authors showed that the GSW scheme satisfies the notion of SRL security if the LWE problem is hard. For completeness, we recall the theorem statement in the following.

Theorem 2.11 (SRL Security of GSW) *If the LWE assumption holds, then the GSW encryption scheme satisfies the notion of SRL security.*

If SRL security is retained in the presence of a key cycle, then we say that the scheme satisfies the notion of 2-circular SRL security. Specifically, we define this notion as above, except that the challenge ciphertext c encrypts $m_b \| \tilde{\mathsf{sk}}$ and the distinguisher is additionally given $\mathsf{Enc}(\tilde{\mathsf{pk}}, \mathsf{sk})$, where $(\tilde{\mathsf{sk}}, \tilde{\mathsf{pk}})$ is an independently sampled key pair of a (possibly different) public-key encryption scheme.

3 Packed Encryption from LWE

In the following we describe a packed version of the dual-Regev encryption scheme [GPV08]. We denote by $n = n(\lambda)$ the lattice dimension, by $q = q(\lambda)$ the modulus (which we assume for simplicity to be even), and by $k = k(\lambda)$ the expansion factor.

$\underline{\mathsf{KeyGen}(1^\lambda, 1^k)}$**:** Sample a uniform $k \times n$ matrix $\mathbf{B} \leftarrow\!\!\$\ \mathbb{Z}_q^{k \times n}$ and a key pair of a regular public-key encryption scheme $(\mathsf{sk}_{\mathsf{PKE}}, \mathsf{pk}_{\mathsf{PKE}}) \leftarrow\!\!\$\ \mathsf{PKE.KeyGen}(1^\lambda)$. The public key consists of $(\mathbf{B}, \mathsf{pk}_{\mathsf{PKE}})$ and the secret key is set to $\mathsf{sk}_{\mathsf{PKE}}$.

$\underline{\mathsf{Enc}(\mathsf{pk}, \mathbf{m})}$**:** To encrypt a k-bit message $\mathbf{m} \in \{0,1\}^k$, sample a uniform randomness vector $\mathbf{r} \leftarrow\!\!\$\ \mathbb{Z}_q^n$ a noise vector $\mathbf{e} \leftarrow\!\!\$\ \chi^k$ and return the ciphertext

$$\mathbf{c} = \left(\mathsf{PKE.Enc}(\mathsf{pk}_{\mathsf{PKE}}, \mathbf{r}), \mathbf{B} \cdot \mathbf{r} + q/2 \cdot \mathbf{m} + \mathbf{e}\right).$$

$\underline{\mathsf{Dec}(\mathsf{sk}, \mathbf{c})}$**:** Parse $\mathbf{c}$ as $(c_{\mathsf{PKE}}, c_1, \dots, c_k)$ and recover the random coins $\mathbf{r} = \mathsf{PKE.Dec}(\mathsf{sk}_{\mathsf{PKE}}, c_{\mathsf{PKE}})$. Let $\mathbf{b}_i$ be the i-th row of $\mathbf{B}$. For $i = 1 \dots k$, compute $m_i = \mathsf{MSB}(c_i - \mathbf{b}_i \cdot \mathbf{r})$, where MSB returns the most significant bit of the input integer. Output $\mathbf{m} = (m_1, \dots, m_k)$.

Clearly, the scheme is perfectly correct since

$$\begin{aligned}(\mathsf{MSB}(c_1 - \mathbf{b}_1 \cdot \mathbf{r}), \dots, \mathsf{MSB}(c_k - \mathbf{b}_k \cdot \mathbf{r})) &= (\mathsf{MSB}(q/2 \cdot m_1 + e_1), \dots, \mathsf{MSB}(q/2 \cdot m_k + e_k))\\ &= (\mathsf{MSB}(q/2 \cdot m_1), \dots, \mathsf{MSB}(q/2 \cdot m_k))\\ &= (m_1, \dots, m_k)\\ &= \mathbf{m}.\end{aligned}$$

Extended Encryption. It is not hard to see that the scheme presented above is (bounded) additively homomorphic over $\mathbb{Z}_q^k$. To lift the class of computable functions to all linear functions, we adopt the standard trick of encrypting the message multiplied by all powers of two $(1, 2, \dots, 2^{\log(q)})$. For convenience, we define the following augmented encryption algorithm.

$\underline{\mathsf{ExtEnc}(\mathsf{pk}, \mathbf{m})}$**:** On input an ℓ-dimensional message $\mathbf{m} \in \mathbb{Z}_q^\ell$, define $\mathbf{m}^{(i,j)}$ as the k-dimensional vector $(0, \ldots, 0, m_j, 0, \ldots, 0)^T$ that contains m_j in the i-th position and 0 everywhere else. Define

$$\mathbf{M} = \begin{bmatrix} \mathbf{m}^{(1,1)} & \mathbf{m}^{(1,1)} \cdot 2 & \ldots & 0^k \\ 0^k & 0^k & \ldots & 0^k \\ \vdots & \vdots & \ddots & \vdots \\ 0^k & 0^k & \ldots & \mathbf{m}^{(k,\ell)} \cdot 2^{\log(q)} \end{bmatrix} \in \mathbb{Z}_q^{k \times \ell \cdot k \cdot \log(q)}.$$

Sample a uniform randomness matrix $\mathbf{R} \leftarrow_\$ \mathbb{Z}_q^{n \times \ell \cdot k \cdot \log(q)}$ and a uniform noise matrix $\mathbf{E} \leftarrow_\$ \chi^{k \times \ell \cdot k \cdot \log(q)}$. Compute

$$\mathbf{C} = \mathbf{B} \cdot \mathbf{R} + \mathbf{M} + \mathbf{E}$$

and return the ciphertext $(\mathsf{PKE.Enc}(\mathsf{pk}_{\mathsf{PKE}}, \mathbf{R}), \mathbf{C})$.

Decryption works, as before, by recovering $\mathbf{R}$ from the public-key encryption scheme and then decrypting $\mathbf{m}$ component-wise.

Almost-Everywhere Dense Encryption. For convenience, we also define an alternative encryption algorithm in the following. Note that the encryption algorithm does not take as input any message, instead it encrypts a uniform k-bit binary vector.

$\underline{\mathsf{DenseEnc}(\mathsf{pk})}$**:** Sample a uniform randomness vector $\mathbf{r} \leftarrow_\$ \mathbb{Z}_q^n$ and return the ciphertext

$$\mathbf{c} = (c_{\mathsf{PKE}}, c_1, \ldots, c_k) = (\mathsf{PKE.Enc}(\mathsf{pk}_{\mathsf{PKE}}, \mathbf{r}), \mathbf{B} \cdot \mathbf{r} + \mathbf{u}) \,.$$

where $\mathbf{u} \leftarrow_\$ \mathbb{Z}_q^k$.

We highlight two facts about this algorithm that are going to be important for our later construction: (i) The decryption algorithm works for both Enc and $\mathsf{DenseEnc}$ algorithms, where the plaintext of $\mathsf{DenseEnc}$ corresponds to $(\mathsf{MSB}(u_1), \ldots, \mathsf{MSB}(u_k))$. In fact, the scheme satisfies perfect correctness in both cases. (ii) The domain of the elements $(c_1, \ldots, c_k)$ is *dense*, i.e. the support of the scheme spans the entire vector space $\mathbb{Z}_q^k$. Since the element c_{PKE} is small (i.e. independent of k) for an appropriate choice of the public-key encryption scheme, we refer to such a property as *almost-everywhere* density.

3.1 Analysis

Here we argue that our scheme as described above satisfies a few properties of interest and we discuss some suitable instantiations for the underlying building blocks.

Semantic Security. First we argue that the scheme satisfies a strong form of semantic security, i.e. the honestly computed ciphertexts are computationally indistinguishable from uniform vectors in $\mathbb{Z}_q^k$. Semantic security for the extended encryption ExtEnc and the dense encryption $\mathsf{DenseEnc}$ follows along the same lines.

Theorem 3.1 (Semantic Security) *If* $(\mathsf{PKE.KeyGen}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec})$ *is semantically secure and the LWE assumption holds, then for all* $\lambda \in \mathbb{N}$ *and all messages* m *it holds that the following distributions are computationally indistinguishable*

$$(\mathsf{pk}, \mathsf{Enc}(\mathsf{pk}, m)) \approx (\mathsf{pk}, \mathsf{PKE.Enc}(\mathsf{pk}_{\mathsf{PKE}}, \mathbf{z}), \mathbf{u}).$$

where $(\mathsf{sk}, \mathsf{pk}) \leftarrow_\$ \mathsf{KeyGen}(1^\lambda, 1^k)$, $\mathbf{z} \leftarrow_\$ \mathbb{Z}_q^n$, *and* $\mathbf{u} \leftarrow_\$ \mathbb{Z}_q^k$.

Proof: The security of the scheme follows routinely by an invocation of semantic security of the public-key encryption scheme and an invocation of the LWE assumption. □

Circuit Privacy. We require that the underlying public-key encryption scheme supports the homomorphic evaluation of linear functions, however we pose no compactness requirements for the evaluated ciphertexts. Instead, we require that the scheme satisfies the following notion of circuit privacy.

Definition 3.2 (Circuit Privacy) *A public-key encryption scheme* $(\mathsf{PKE.KeyGen}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec})$ *is circuit-private for the circuit class* $\{\mathfrak{C}_\lambda\}_{\lambda\in\mathbb{N}}$ *if for all* $\Pi \in \mathfrak{C}_\lambda$*, all* $(\mathsf{sk}_{\mathsf{PKE}}, \mathsf{pk}_{\mathsf{PKE}})$ *in the support of* $\mathsf{PKE.KeyGen}$*, and all messages* m*, it holds that the following distributions are statistically indistinguishable*

$$(\mathsf{pk}, \mathsf{PKE.Enc}(\mathsf{pk}_{\mathsf{PKE}}, \Pi(m))) \approx (\mathsf{pk}, \mathsf{PKE.Eval}(\mathsf{pk}_{\mathsf{PKE}}, \Pi, \mathsf{PKE.Enc}(\mathsf{pk}_{\mathsf{PKE}}, m))).$$

Many fully-homomorphic encryption schemes are known to satisfy such a notion (see e.g. [BdMW16, DS16, OPP14]). As we do not require compactness, we can even instantiate this building block using a two-round oblivious transfer with statistical sender privacy [PVW08, BD18, DGI+19, BDGM19] and information-theoretic garbled circuits [Kil88, AIK04, IP07].

4 Constructing XiO

In the following we present the construction of XiO from the GSW scheme ($\mathsf{GSW.KeyGen}, \mathsf{GSW.Enc}, \mathsf{GSW.Eval}, \mathsf{GSW.Dec}$) and the packed version of the dual-Regev encryption ($\mathsf{dR.KeyGen}, \mathsf{dR.Enc}, \mathsf{dR.Eval}, \mathsf{dR.Dec}$) as described in Section 3. The construction and the analysis is largely unchanged from [GP20] (which in turn is based on the blueprint of [BDGM20]), except that we include some extra elements in the XiO scheme to account for the fact that some parts of the dual-Regev ciphertext are not dense.

4.1 Construction

The scheme assumes a long uniform string that is, for convenience, split in two chunks:

- A sequence of randomization vectors $(\mathbf{r}^*_1, \ldots, \mathbf{r}^*_{\eta-\log(k)})$ for the GSW scheme $\mathsf{GSW.PubCoin}$, where each $\mathbf{r}_i \in [-\tilde{B}, +\tilde{B}]^{m\cdot k}$.
- A sequence of dense ciphertexts $((h_{1,1}, \ldots, h_{1,k}), \ldots, (h_{\eta-\log(k),1}, \ldots, h_{\eta-\log(k),k}))$ for packed dual-Regev scheme $\mathsf{dR.PubCoin}$, where each $(h_{i,1}, \ldots, h_{i,k}) \in \mathbb{Z}_q^k$.

On input the security parameter 1^λ and the circuit $\Pi : \{0,1\}^\eta \to \{0,1\}$, the obfuscator proceeds as follows.

Setting the Public Keys: Sample a dual-Regev key pair $(\bar{\mathsf{sk}}, \bar{\mathsf{pk}}) \leftarrow\!\!\$\ \mathsf{dR.KeyGen}(1^\lambda, 1^k)$ and GSW key pair $(\mathsf{sk}, \mathsf{pk}) \leftarrow\!\!\$\ \mathsf{GSW.KeyGen}(1^\lambda; q)$, where q is the modulus defined by the dual-Regev scheme. Compute a GSW encryption $\mathbf{c}_\Pi \leftarrow \mathsf{GSWEnc}(\mathsf{pk}, \Pi)$ of the binary representation of the circuit Π.

Compute a Key Cycle: Compute a GSW encryption of the dual-Regev secret key $\mathbf{c}_{\bar{\mathsf{sk}}} = \mathsf{GSW.Enc}(\mathsf{pk}, \bar{\mathsf{sk}})$ and a dual-Regev extended encryption of the GSW secret key

$$\left(\bar{c}_{\mathsf{sk}}, \bar{\mathbf{C}}_{\mathsf{sk}}\right) = \mathsf{dR.ExtEnc}(\bar{\mathsf{pk}}, \mathsf{sk}; \mathbf{S}).$$

where $\mathsf{sk} \in \mathbb{Z}_q^{n+1}$ and $\mathbf{S} \leftarrow\!\!\$\ \mathbb{Z}_q^{n\times\log(q)\cdot k\cdot(n+1)}$.

Decryption Hints: For all indices $i \in \{0,1\}^{\eta-\log(k)}$, do the following.

Evaluate the Circuit: Let $\Phi_i : \{0,1\}^{|\Pi|} \to \{0,1\}^k$ be the universal circuit that, on input a circuit description Π, returns the i-th block (where each block consists of k bits) of the corresponding truth table. Compute

$$\mathbf{c}_i \leftarrow \mathsf{GSW.Eval}(\mathsf{pk}, \Phi_i, \mathbf{c}_\Pi).$$

Compute the Encryption Header: Sample a uniform $\mathbf{r}_i \leftarrow\!\!\$\ \mathbb{Z}_q^n$ and set $h_{i,0} = \mathsf{PKE.Enc}(\mathsf{pk}_{\mathsf{PKE}}, \mathbf{r}_i)$, where $\mathsf{pk}_{\mathsf{PKE}}$ is the public-key encryption scheme defined by the key generation of dual-Regev.

Compute the Low-Order Bits: Parse the i-th block of $\mathsf{dR.PubCoin}$ as

$$(h_{i,1}, \ldots, h_{i,k}) = \mathbf{B} \cdot \mathbf{r}_i + (u_{i,1}, \ldots, u_{i,k}) \in \mathbb{Z}_q^k$$

for some $(u_{i,1}, \ldots, u_{i,k}) \in \mathbb{Z}_q^k$. Let $\Psi_i : \{0,1\}^\lambda \to \{0,1\}^k$ be circuit that, on input the dual-Regev secret key, computes the decryption of $(h_{i,0}, h_{i,1}, \ldots, h_{i,k})$, i.e.

$$\Psi_i(\bar{\mathsf{sk}}) = \mathsf{dR.Dec}(\bar{\mathsf{sk}}, (h_{i,0}, h_{i,1}, \ldots, h_{i,k})).$$

Compute homomorphically the k-bit ciphertext $\mathbf{c}_{i,\mathsf{MSB}} = \mathsf{GSW.Eval}(\mathsf{pk}, \Psi_i, \mathbf{c}_{\bar{\mathsf{sk}}})$.

Rerandomize the Ciphertext: Parse the i-th block of $\mathsf{GSW.PubCoin}$ as $\mathbf{r}_i^* \in [-\tilde{B}, +\tilde{B}]^{m \cdot k}$ and compute

$$\mathbf{c}'_{i,\mathsf{MSB}} = \mathbf{c}_{i,\mathsf{MSB}} + \mathsf{GSW.ColEnc}(\mathsf{pk}, 0^k; \mathbf{r}_i^*).$$

Proxy Re-Encrypt: Define $\mathbf{d}_i$ as the GSW ciphertext resulting from the homomorphic sum of $\mathbf{c}'_{i,\mathsf{MSB}}$ and $\mathbf{c}_i$, i.e. $\mathbf{d}_i = \mathbf{c}'_{i,\mathsf{MSB}} + \mathbf{c}_i$. Observe that $\mathbf{d}_i$ is a k-bit ciphertext and let $\boldsymbol{\ell}_{i,j}$ be the linear function associated with the decryption of the j-th bit. Define $\boldsymbol{\ell}_i = (\boldsymbol{\ell}_{i,1}, \ldots, \boldsymbol{\ell}_{i,k})$ and compute

$$\bar{\mathbf{c}}_i = \bar{\mathbf{C}}_{\mathsf{sk}} \cdot \mathsf{Bit}(\boldsymbol{\ell}_i) + (h_{i,1}, \ldots, h_{i,k}) \in \mathbb{Z}_q^k$$

where the function $\mathsf{Bit} : \mathbb{Z}_q^{k \cdot (n+1)} \to \{0,1\}^{\log(q) \cdot k \cdot (n+1)}$ is the bit decomposition operator.

Release Hint: Compute the i-th decryption hint as

$$\boldsymbol{\rho}_i = \mathbf{S} \cdot \mathsf{Bit}(\boldsymbol{\ell}_i) + \mathbf{r}_i \in \mathbb{Z}_q^n.$$

Output: The obfuscated circuit consists of the public keys $(\mathsf{pk}, \bar{\mathsf{pk}})$, the key cycle $(\bar{\mathbf{C}}_{\mathsf{sk}}, \mathbf{c}_{\bar{\mathsf{sk}}})$, the GSW encryption of the circuit $\mathbf{c}_\Pi$, the encryption headers $(h_1, \ldots, h_{\eta - \log(k)})$, and the decryption hints $(\boldsymbol{\rho}_1, \ldots, \boldsymbol{\rho}_{\eta - \log(k)})$.

To evaluate the obfuscated circuit on input x, let i be the index of the block of the truth table of Π that contains $\Pi(x)$. The evaluator computes $\bar{\mathbf{c}}_i$ as specified above (note that all the operations are public, given the information included in the obfuscated circuit) and recovers $\Pi^{(i)}$ (the i-th block of the truth table of Π) by computing

$$\Pi^{(i)} = \mathsf{MSB}(\mathbf{c}_i - \mathbf{B} \cdot \boldsymbol{\rho}_i)$$

where MSB returns the most significant bit of each component of the input vector.

Correctness. To see why the evaluation algorithm is correct, recall that

$$\bar{\mathbf{c}}_i = \bar{\mathbf{C}}_{\mathsf{sk}} \cdot \mathsf{Bit}(\boldsymbol{\ell}_i) + (h_{i,1}, \ldots, h_{i,k}).$$

First observe that $(h_{i,0}, h_{i,1}, \ldots, h_{i,k})$ is a ciphertext in the support of $\mathsf{dR.DenseEnc}(\bar{\mathsf{pk}})$, and in particular

$$\begin{aligned}(h_{i,0}, h_{i,1}, \ldots, h_{i,k}) &= (\mathsf{PKE.Enc}(\mathsf{pk}_{\mathsf{PKE}}, \mathbf{r}_i), \mathbf{B} \cdot \mathbf{r}_i + (u_{i,1}, \ldots, u_{i,k})) \\ &= (\mathsf{PKE.Enc}(\mathsf{pk}_{\mathsf{PKE}}, \mathbf{r}_i), \mathbf{B} \cdot \mathbf{r}_i + \mathbf{u}_i)\,.\end{aligned}$$

Furthermore, observe that

$$\begin{aligned}\mathbf{d}_i &= \mathbf{c}'_{i,\mathsf{MSB}} + \mathbf{c}_i \\ &= \mathbf{c}'_{i,\mathsf{MSB}} + \mathsf{GSW.ColEnc}(\mathsf{pk}, \Pi^{(i)}) \\ &= \mathsf{GSW.ColEnc}(\mathsf{pk}, (\mathsf{MSB}(u_1), \ldots, \mathsf{MSB}(u_k))) + \mathsf{GSW.ColEnc}(\mathsf{pk}, \Pi^{(i)}) \\ &= \mathsf{GSW.ColEnc}(\mathsf{pk}, (\mathsf{MSB}(u_1), \ldots, \mathsf{MSB}(u_k)) \oplus \Pi^{(i)}) \\ &= \mathsf{GSW.ColEnc}(\mathsf{pk}, (\mathsf{MSB}(u_1) \oplus \Pi_1^{(i)}, \ldots, \mathsf{MSB}(u_k) \oplus \Pi_k^{(i)}))\end{aligned}$$

by the evaluation correctness of GSW. By the almost-linear decryption of GSW, it follows that

$$\bar{\mathbf{C}}_{\mathsf{sk}} \cdot \mathsf{Bit}(\boldsymbol{\ell}_i) = \mathbf{B} \cdot \tilde{\mathbf{s}}_i + \boldsymbol{\xi}_i + \boldsymbol{\zeta}_i + q/2 \cdot \left(\mathsf{MSB}(u_1) \oplus \Pi_1^{(i)}, \ldots, \mathsf{MSB}(u_k) \oplus \Pi_k^{(i)}\right)$$

where $\boldsymbol{\xi}_i$ is the decryption noise of the packed dual-Regev scheme (i.e. the subset sum of the noise terms of $\bar{\mathbf{C}}_{\mathsf{sk}}$) and $\boldsymbol{\zeta}_i$ is the decryption noise of the GSW ciphertext. It follows that $\|\boldsymbol{\xi}_i\|_\infty \leq B \cdot \log(q) \cdot k \cdot (n+1)$ and, by Lemma 2.1, $\|\boldsymbol{\zeta}_i\|_\infty \leq \tilde{B}$ with all but negligible probability. Note that, by linearity we have that $\tilde{\mathbf{s}}_i = \mathbf{S} \cdot \mathsf{Bit}(\boldsymbol{\ell}_i)$. Consequently, it holds that

$$\begin{aligned}
\bar{\mathbf{c}}_i &= \mathbf{B} \cdot \tilde{\mathbf{s}}_i + \boldsymbol{\xi}_i + \boldsymbol{\zeta}_i + q/2 \cdot \left(\mathsf{MSB}(u_1) \oplus \Pi_1^{(i)}, \ldots, \mathsf{MSB}(u_k) \oplus \Pi_k^{(i)}\right) + \mathbf{B} \cdot \mathbf{r}_i + \mathbf{u}_i \\
&= \mathbf{B} \cdot (\tilde{\mathbf{s}}_i + \mathbf{r}_i) + \boldsymbol{\xi}_i + \boldsymbol{\zeta}_i + q/2 \cdot \left(\mathsf{MSB}(u_1) \oplus \Pi_1^{(i)}, \ldots, \mathsf{MSB}(u_k) \oplus \Pi_k^{(i)}\right) + \mathbf{u}_i \\
&= \mathbf{B} \cdot (\tilde{\mathbf{s}}_i + \mathbf{r}_i) + q/2 \cdot \Pi^{(i)} + \mathbf{v}_i \\
&= \mathbf{B} \cdot \boldsymbol{\rho}_i + q/2 \cdot \Pi^{(i)} + \mathbf{v}_i
\end{aligned}$$

where $\mathbf{v}_i = \mathbf{u}_i + q/2 \cdot \mathsf{MSB}(\mathbf{u}_i) + \boldsymbol{\xi}_i + \boldsymbol{\zeta}_i$ and $\|\mathbf{v}_i\|_\infty < q/4$ with all but negligible probability, over the random choice of $\mathbf{u}_i$. This is because $\mathbf{d}_i$ is statistically close to a fresh GSW encryption of $(\mathsf{MSB}(u_1), \ldots, \mathsf{MSB}(u_k)) \oplus \Pi^{(i)}$, by Lemma 2.9. Therefore we have that

$$\begin{aligned}
\mathsf{MSB}\left(\mathbf{c}_i - \mathbf{B} \cdot \boldsymbol{\rho}_i\right) &= \mathsf{MSB}\left(\mathbf{B} \cdot \boldsymbol{\rho}_i + q/2 \cdot \Pi^{(i)} + \mathbf{v}_i - \mathbf{B} \cdot \boldsymbol{\rho}_i\right) \\
&= \mathsf{MSB}\left(q/2 \cdot \Pi^{(i)} + \mathbf{v}_i\right) \\
&= \mathsf{MSB}\left(q/2 \cdot \Pi^{(i)}\right) \\
&= \Pi^{(i)}
\end{aligned}$$

with the same probability.

4.2 Analysis

In the following we show that our XiO scheme satisfies the notion of security for indistinguishability obfuscation.

Theorem 4.1 (XiO Security) *If the GSW scheme* $(\mathsf{GSW.KeyGen}, \mathsf{GSW.Enc}, \mathsf{GSW.Eval}, \mathsf{GSW.Dec})$ *and the packed dual-Regev scheme* $(\mathsf{dR.KeyGen}, \mathsf{dR.Enc}, \mathsf{dR.Eval}, \mathsf{dR.Dec})$ *are 2-circular SRL secure, then the XiO scheme as described above is secure.*

Proof: We prove the scheme via a series of hybrid experiments. The proof follows closely the argument of [GP20] and it is reported here for completeness.

- Hybrid $\mathcal{H}_0$: This is the original obfuscation of the circuit Π_0.
- Hybrid $\mathcal{H}_1$: This hybrid is identical to the previous one, except that for all $i \in \{0,1\}^{\eta - \log(k)}$ we sample $\mathbf{c}'_{i,\mathsf{MSB}}$ as

  $$\mathbf{c}'_{i,\mathsf{MSB}} = \mathsf{GSW.ColEnc}(\mathsf{pk}, (\mathsf{MSB}(u_{i,1}), \ldots, \mathsf{MSB}(u_{i,k})); \mathbf{r}_i^*)$$

  where $\mathbf{r}_i^* \leftarrow\!\!\$\, [-\tilde{B}, +\tilde{B}]^{m \cdot k}$. Let $\mathbf{r}_{\Psi,i}$ be the random coins of $\mathbf{c}_{i,\mathsf{MSB}}$ (as computed in the original protocol). We additionally set the i-the block of the $\mathsf{GSW.PubCoin}$ to $\mathbf{r}_i^* - \mathbf{r}_{\Psi,i}$. Statistical indistinguishability with respect to the previous hybrid follows from an invocation of Lemma 2.9.

- Hybrid $\mathcal{H}_2$: This hybrid is identical to the previous one, except that for all $i \in \{0,1\}^{\eta-\log(k)}$ we set
$$(h_{i,1}, \ldots, h_{i,k}) = \mathbf{B} \cdot \mathbf{r}_i + (u_{i,1}, \ldots, u_{i,k})$$
where $(u_{i,1}, \ldots, u_{i,k}) \leftarrow_{\$} \mathbb{Z}_q^k$. The only difference with respect to the previous hybrid is that the obfuscator knows the values $(u_{i,1}, \ldots, u_{i,k})$ ahead of time. However, the two distributions are identical to the eyes of the distinguisher and therefore the change here is only syntactical.

- Hybrid $\mathcal{H}_3$: In this hybrid we generate, for all $i \in \{0,1\}^{\eta-\log(k)}$, $\bar{\mathbf{c}}_i$ as follows
$$\bar{\mathbf{c}}_i = \mathbf{B} \cdot \mathbf{t}_i + \boldsymbol{\xi}_i + \boldsymbol{\zeta}_i + q/2 \cdot \left(\mathsf{MSB}(u_1) \oplus \Pi_1^{(i)}, \ldots, \mathsf{MSB}(u_k) \oplus \Pi_k^{(i)}\right) + \mathbf{u}_i$$
where $\mathbf{t}_i \leftarrow_{\$} \mathbb{Z}_q^n$. Here $\boldsymbol{\xi}_i$ and $\boldsymbol{\zeta}_i$ denote the decryption noises of $\bar{\mathbf{C}}_{\mathsf{sk}}$ and $\mathbf{d}_i$, respectively. Furthermore, we set $(h_{i,1}, \ldots, h_{i,k}) = \bar{\mathbf{c}}_i - \bar{\mathbf{C}}_{\mathsf{sk}} \cdot \mathsf{Bit}(\boldsymbol{\ell}_i)$ and $h_{i,0} = \mathsf{PKE.Enc}(\mathsf{pk}_{\mathsf{PKE}}, \mathbf{t}_i - \mathbf{S} \cdot \mathsf{Bit}(\boldsymbol{\ell}_i))$. Note that $\mathbf{c}'_{i,\mathsf{MSB}}$ is fixed (and in particular is independent of $h_{i,0}$) and thus the above variables are always well defined. In fact, observe that
$$\begin{aligned}(h_{i,1}, \ldots, h_{i,k}) &= \bar{\mathbf{c}}_i - \bar{\mathbf{C}}_{\mathsf{sk}} \cdot \mathsf{Bit}(\boldsymbol{\ell}_i) \\ &= \bar{\mathbf{c}}_i - \mathbf{B} \cdot (\mathbf{S} \cdot \mathsf{Bit}(\boldsymbol{\ell}_i)) - \boldsymbol{\xi}_i - \boldsymbol{\zeta}_i - q/2 \cdot \left(\mathsf{MSB}(u_1) \oplus \Pi_1^{(i)}, \ldots, \mathsf{MSB}(u_k) \oplus \Pi_k^{(i)}\right) \\ &= \mathbf{B} \cdot (\mathbf{t}_i - \mathbf{S} \cdot \mathsf{Bit}(\boldsymbol{\ell}_i)) + \mathbf{u}_i\end{aligned}$$
which is exactly the same distribution is in the previous hybrid. Furthermore, note that we now have $\boldsymbol{\rho}_i = \boldsymbol{t}_i$. Thus the change introduced here is only syntactical.

- Hybrid $\mathcal{H}_4$: In this hybrid we generate, for all $i \in \{0,1\}^{\eta-\log(k)}$, $\bar{\mathbf{c}}_i$ as a fresh encryption of $\Pi^{(i)}$, i.e.
$$\bar{\mathbf{c}}_i = \mathbf{B} \cdot \mathbf{t}_i + q/2 \cdot \Pi^{(i)} + \mathbf{w}_i$$
where $\mathbf{t}_i \leftarrow_{\$} \mathbb{Z}_q^n$ and $\mathbf{w}_i \leftarrow_{\$} [-q/4, +q/4]^k$. Note that we can bound $\|\boldsymbol{\xi}_i\|_\infty \leq B \cdot \log(q) \cdot k \cdot (n+1)$. Furthermore, we have that $\|\boldsymbol{\zeta}_i\|_\infty \leq \tilde{B}$ with all but negligible probability over the random choice of $\mathbf{r}_i^*$, by Lemma 2.1. Statistical indistinguishability follows from another application of Lemma 2.1.

- Hybrid $\mathcal{H}_5$: Here we define, for all $i \in \{0,1\}^{\eta-\log(k)}$, the circuit $\Gamma_i : \mathbb{Z}_q^{n \times (n+1) \cdot k \cdot \log(q)} \to \mathbb{Z}_q^n$ as the following circuit
$$\Gamma_i(\mathbf{X}) = \mathbf{t}_i - \mathbf{X} \cdot \mathsf{Bit}(\boldsymbol{\ell}_i).$$
Then we set $h_{i,0} = \mathsf{PKE.Eval}(\mathsf{pk}_{\mathsf{PKE}}, \Gamma_i, \bar{c}_{\mathsf{sk}})$. To see why the hybrids are statistically close, recall that $\bar{c}_{\mathsf{sk}} = \mathsf{PKE.Enc}(\mathsf{pk}_{\mathsf{PKE}}, \mathbf{S})$, where $\mathbf{S}$ are the random coins used in the encryption of sk. Then
$$\mathsf{PKE.Eval}(\mathsf{pk}_{\mathsf{PKE}}, \Gamma_i, \mathsf{PKE.Enc}(\mathsf{pk}_{\mathsf{PKE}}, \mathbf{S})) \approx \mathsf{PKE.Enc}(\mathsf{pk}_{\mathsf{PKE}}, \mathbf{t}_i - \mathbf{S} \cdot \mathsf{Bit}(\boldsymbol{\ell}_i))$$
by the statistical circuit privacy of the public-key encryption scheme. Jumping ahead to our modified scheme, note that the same hybrid step would have succeeded (with the same argument) if we were given a GSW encryption of $\mathbf{S}$.

- Hybrid $\mathcal{H}_6$: In this hybrid we compute $\mathbf{c}_\Pi$ as an encryption of Π_1 instead of Π_0. Note that we no longer use the secret key of either of the encryption schemes to obfuscate the circuit, besides the encrypted key cycle. The random coins $\mathsf{GSW.PubCoin}$ (which we set to $\mathbf{r}_i^* - \mathbf{r}_{\Psi,i}$) can be modelled as an SRL leakage and therefore security follows from a standard reduction to the 2-circular SRL security of GSW and packed dual-Regev: The experiment is defined with respect to the set of circuits $(\Psi_1, \ldots, \Psi_{\eta-\log(k)})$ (as defined in $\mathcal{H}_4$) and the reduction uses the extra ciphertexts $(\mathbf{c}_1^*, \ldots, \mathbf{c}_{\eta-\log(k)}^*)$ to compute $\mathbf{c}'_{i,\mathsf{MSB}}$ to be $\mathbf{c}_i^* + (0^n, \mathsf{MSB}(u_{i,1}), \ldots, 0^n, \mathsf{MSB}(u_{i,k}))$. The rest of the information is already given by the SRL experiment or recomputed as in $\mathcal{H}_4$.

- Hybrids $\mathcal{H}_7 \ldots \mathcal{H}_{11}$: In this series of hybrids we undo all changes that we did in $\mathcal{H}_5 \ldots \mathcal{H}_1$. The statistical indistinguishability follows by the same arguments as before. Note that $\mathcal{H}_{11}$ is the original obfuscation of Π_1. This concludes our proof.

□

4.3 Parameters

The analysis of our scheme sets two constraints on the noise growth of the encryption schemes. The first application of the smudging Lemma requires that the noise bound $\tilde{B}$ is exponentially larger than the noise bound on evaluated GSW ciphertexts $\hat{B}$, i.e. (1) $\tilde{B} \geq 2^{\lambda} \cdot \hat{B}$. The second application requires that (2) $q/4 \geq 2^{\lambda} \cdot \tilde{B}$ and (3) $q/4 \geq 2^{\lambda} \cdot B \cdot \log(q) \cdot k \cdot (n+1)$, to ensure that the term u_i (minus its most significant bit) properly floods the noise terms of the GSW and dual-Regev ciphertexts. Note that, for an appropriate choice of parameters of the GSW scheme, condition (3) is already implied by conditions (1) and (2) and therefore all we need ensures is that these two conditions are satisfiable. It is not hard to see that the above pair of constraints can be satisfied by setting the modulo-to-noise ratio of the LWE assumption to be super-polynomial.

To make the XiO scheme compressing, we set $k = 2^{\eta/4}$. We analyze the size of each component of the obfuscated circuit in the following:

- The size of the public keys $(\mathsf{pk}, \bar{\mathsf{pk}})$ is linear in k and thus can be bounded by $2^{\eta/4} \cdot \mathsf{poly}(\lambda)$.
- The size of the key cycle $(\bar{\mathbf{C}}_{\mathsf{sk}}, \mathbf{c}_{\bar{\mathsf{sk}}})$ can be bounded by $k^2 \cdot \mathsf{poly}(\lambda) = 2^{\eta/2} \cdot \mathsf{poly}(\lambda)$.
- The GSW encryption of the circuit $\mathbf{c}_{\Pi}$ is of size $|\Pi| \cdot \mathsf{poly}(\lambda)$.
- The size of an encryption header h_i is $\mathsf{poly}(\lambda)$ (and in particular independent of k by the almost-everywhere density of dual-Regev) and the size of each decryption hint $\boldsymbol{\rho}_i$ is also $\mathsf{poly}(\lambda)$ (by the randomness succinctness of dual-Regev). It follows that the total size of the encryption headers and the decryption hints can be bounded by $2^{\eta - \log(k)} \cdot \mathsf{poly}(\lambda) = 2^{3\eta/4} \cdot \mathsf{poly}(\lambda)$.

The summation of the above terms is sublinear in $\mathsf{poly}(|\Pi|, \lambda) \cdot 2^{\eta(1-\varepsilon)}$ (for $\varepsilon > 0$) and therefore the XiO scheme satisfies non-trivial efficiency. Note that we omitted the public parameters $\mathsf{GSW.PubCoin}$ and $\mathsf{dR.PubCoin}$ from the analysis since they are uniformly at random and thus can be computed in the pre-processing stage of the obfuscator.

4.4 Randomness-Key Circularity

As it is described above, our scheme assumes the 2-circular SRL security of GSW and the variant of packed dual-Regev (presented in Section 3). While SRL security can be based on the plain LWE assumption for GSW alone, we conjecture that it is retained also in the presence of a 2-key cycle.

It is instructive to discuss why a key cycle is needed: On the one hand we need to key-switch the GSW encryption into a packed dual-Regev ciphertext, which has succinct randomness and therefore short decryption hints. This requires us to encrypt the GSW secret key under the dual-Regev encryption. On the other hand, we also need to recover the most significant bits of the vector $\mathbf{u}_i$ (component-wise) to make sure that the smudging noise does not interfere with the correctness of the scheme. This is done by parsing $(h_{i,0}, h_{i,1}, \ldots, h_{i,k})$ as a dense ciphertext of packed dual-Regev and computing the decryption homomorphically (thus the need to encrypt the dual-Regev secret key under GSW).

We observe that we can slightly tweak the XiO scheme to weaken the circularity assumption. Recall that the packed dual-Regev decryption works by decrypting the randomness $\mathbf{r}_i$ from the term $h_{i,0}$ and uses it to recover the message. Our idea is to bypass the first step by simply including in the obfuscated circuit a GSW encryption of the randomnesses $(\mathbf{r}_1, \ldots, \mathbf{r}_{\eta - \log(q)})$. With this modification we can omit from the obfuscated circuit both the GSW encryption of the secret key $\mathsf{GSW.Enc}(\mathsf{pk}, \bar{\mathsf{sk}})$ and the encryption headers $(h_{1,0}, \ldots, h_{\eta-\log(q),0})$. However, we need to add a GSW encryption of $\mathbf{S}$ (the randomness used to compute $\bar{\mathbf{C}}_{\mathsf{sk}}$) in order to simulate the GSW encryption of $\mathbf{r}_i$ in the proof. Thus, the size of the obfuscated circuit is (asymptotically) identical. The security argument follows along the same lines of what already shown, except that the computational step boils down to a randomness-key circularity assumption. More precisely, we assume that SRL security holds in the presence of the following cycle:

$$\left(\mathsf{GSW.Enc}(\mathsf{pk}, \mathbf{S}), \bar{\mathbf{C}}_{\mathsf{sk}}\right)$$

where $(\bar{c}_{\mathsf{sk}}, \bar{\mathbf{C}}_{\mathsf{sk}}) = \mathsf{dR.ExtEnc}(\bar{\mathsf{pk}}, \mathsf{sk}; \mathbf{S})$. We stress that such an assumption is *strictly weaker* than assuming SRL security of a 2-key cycle, since we could have used the homomorphism of GSW and the randomness recoverability of dual-Regev to compute a GSW encryption of $\mathbf{S}$ homomorphically.

References

[Agr19] Shweta Agrawal. Indistinguishability obfuscation without multilinear maps: New methods for bootstrapping and instantiation. In Yuval Ishai and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2019, Part I*, volume 11476 of *Lecture Notes in Computer Science*, pages 191–225, Darmstadt, Germany, May 19–23, 2019. Springer, Heidelberg, Germany.

[AIK04] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Cryptography in NC^0. In *45th Annual Symposium on Foundations of Computer Science*, pages 166–175, Rome, Italy, October 17–19, 2004. IEEE Computer Society Press.

[AIK11] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. How to garble arithmetic circuits. In Rafail Ostrovsky, editor, *52nd Annual Symposium on Foundations of Computer Science*, pages 120–129, Palm Springs, CA, USA, October 22–25, 2011. IEEE Computer Society Press.

[AJL+12] Gilad Asharov, Abhishek Jain, Adriana López-Alt, Eran Tromer, Vinod Vaikuntanathan, and Daniel Wichs. Multiparty computation with low communication, computation and interaction via threshold FHE. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 483–501, Cambridge, UK, April 15–19, 2012. Springer, Heidelberg, Germany.

[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, *Advances in Cryptology – CRYPTO 2019, Part III*, volume 11694 of *Lecture Notes in Computer Science*, pages 284–332, Santa Barbara, CA, USA, August 18–22, 2019. Springer, Heidelberg, Germany.

[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, *Advances in Cryptology – EUROCRYPT 2017, Part I*, volume 10210 of *Lecture Notes in Computer Science*, pages 152–181, Paris, France, April 30 – May 4, 2017. Springer, Heidelberg, Germany.

[BD18] Zvika Brakerski and Nico Döttling. Two-message statistically sender-private OT from LWE. In Amos Beimel and Stefan Dziembowski, editors, *TCC 2018: 16th Theory of Cryptography Conference, Part II*, volume 11240 of *Lecture Notes in Computer Science*, pages 370–390, Panaji, India, November 11–14, 2018. Springer, Heidelberg, Germany.

[BDGM19] Zvika Brakerski, Nico Döttling, Sanjam Garg, and Giulio Malavolta. Leveraging linear decryption: Rate-1 fully-homomorphic encryption and time-lock puzzles. In Dennis Hofheinz and Alon Rosen, editors, *TCC 2019: 17th Theory of Cryptography Conference, Part II*, volume 11892 of *Lecture Notes in Computer Science*, pages 407–437, Nuremberg, Germany, December 1–5, 2019. Springer, Heidelberg, Germany.

[BDGM20] Zvika Brakerski, Nico Döttling, Sanjam Garg, and Giulio Malavolta. Candidate iO from homomorphic encryption schemes. In Anne Canteaut and Yuval Ishai, editors, *Advances in Cryptology – EUROCRYPT 2020, Part I*, volume 12105 of *Lecture Notes in Computer Science*, pages 79–109, Zagreb, Croatia, May 10–14, 2020. Springer, Heidelberg, Germany.

[BdMW16] Florian Bourse, Rafaël del Pino, Michele Minelli, and Hoeteck Wee. FHE circuit privacy almost for free. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology – CRYPTO 2016, Part II*, volume 9815 of *Lecture Notes in Computer Science*, pages 62–89, Santa Barbara, CA, USA, August 14–18, 2016. Springer, Heidelberg, Germany.

[BGI+01] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In Joe Kilian, editor, *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 1–18, Santa Barbara, CA, USA, August 19–23, 2001. Springer, Heidelberg, Germany.

[BGMZ18] James Bartusek, Jiaxin Guan, Fermi Ma, and Mark Zhandry. Return of GGH15: Provable security against zeroizing attacks. In Amos Beimel and Stefan Dziembowski, editors, *TCC 2018: 16th Theory of Cryptography Conference, Part II*, volume 11240 of *Lecture Notes in Computer Science*, pages 544–574, Panaji, India, November 11–14, 2018. Springer, Heidelberg, Germany.

[BZ14] Dan Boneh and Mark Zhandry. Multiparty key exchange, efficient traitor tracing, and more from indistinguishability obfuscation. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology – CRYPTO 2014, Part I*, volume 8616 of *Lecture Notes in Computer Science*, pages 480–499, Santa Barbara, CA, USA, August 17–21, 2014. Springer, Heidelberg, Germany.

[CGH17] Yilei Chen, Craig Gentry, and Shai Halevi. Cryptanalyses of candidate branching program obfuscators. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *Advances in Cryptology – EUROCRYPT 2017, Part III*, volume 10212 of *Lecture Notes in Computer Science*, pages 278–307, Paris, France, April 30 – May 4, 2017. Springer, Heidelberg, Germany.

[CHL+15] Jung Hee Cheon, Kyoohyung Han, Changmin Lee, Hansol Ryu, and Damien Stehlé. Cryptanalysis of the multilinear map over the integers. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015, Part I*, volume 9056 of *Lecture Notes in Computer Science*, pages 3–12, Sofia, Bulgaria, April 26–30, 2015. Springer, Heidelberg, Germany.

[CLT13] Jean-Sébastien Coron, Tancrède Lepoint, and Mehdi Tibouchi. Practical multilinear maps over the integers. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013, Part I*, volume 8042 of *Lecture Notes in Computer Science*, pages 476–493, Santa Barbara, CA, USA, August 18–22, 2013. Springer, Heidelberg, Germany.

[CVW18] Yilei Chen, Vinod Vaikuntanathan, and Hoeteck Wee. GGH15 beyond permutation branching programs: Proofs, attacks, and candidates. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology – CRYPTO 2018, Part II*, volume 10992 of *Lecture Notes in Computer Science*, pages 577–607, Santa Barbara, CA, USA, August 19–23, 2018. Springer, Heidelberg, Germany.

[DGG+18] Nico Döttling, Sanjam Garg, Divya Gupta, Peihan Miao, and Pratyay Mukherjee. Obfuscation from low noise multilinear maps. In Debrup Chakraborty and Tetsu Iwata, editors, *Progress in Cryptology - INDOCRYPT 2018: 19th International Conference in Cryptology in India*, volume 11356 of *Lecture Notes in Computer Science*, pages 329–352, New Delhi, India, December 9–12, 2018. Springer, Heidelberg, Germany.

[DGI+19] Nico Döttling, Sanjam Garg, Yuval Ishai, Giulio Malavolta, Tamer Mour, and Rafail Ostrovsky. Trapdoor hash functions and their applications. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology – CRYPTO 2019, Part III*, volume 11694 of *Lecture Notes in Computer Science*, pages 3–32, Santa Barbara, CA, USA, August 18–22, 2019. Springer, Heidelberg, Germany.

[DJ01] Ivan Damgård and Mats Jurik. A generalisation, a simplification and some applications of Paillier's probabilistic public-key system. In Kwangjo Kim, editor, *PKC 2001: 4th International*

Workshop on Theory and Practice in Public Key Cryptography, volume 1992 of *Lecture Notes in Computer Science*, pages 119–136, Cheju Island, South Korea, February 13–15, 2001. Springer, Heidelberg, Germany.

[DS16] Léo Ducas and Damien Stehlé. Sanitization of FHE ciphertexts. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology – EUROCRYPT 2016, Part I*, volume 9665 of *Lecture Notes in Computer Science*, pages 294–310, Vienna, Austria, May 8–12, 2016. Springer, Heidelberg, Germany.

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In Michael Mitzenmacher, editor, *41st Annual ACM Symposium on Theory of Computing*, pages 169–178, Bethesda, MD, USA, May 31 – June 2, 2009. ACM Press.

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

[GGH+13b] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *54th Annual Symposium on Foundations of Computer Science*, pages 40–49, Berkeley, CA, USA, October 26–29, 2013. IEEE Computer Society Press.

[GGH15] Craig Gentry, Sergey Gorbunov, and Shai Halevi. Graph-induced multilinear maps from lattices. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015: 12th Theory of Cryptography Conference, Part II*, volume 9015 of *Lecture Notes in Computer Science*, pages 498–527, Warsaw, Poland, March 23–25, 2015. Springer, Heidelberg, Germany.

[GGHR14] Sanjam Garg, Craig Gentry, Shai Halevi, and Mariana Raykova. Two-round secure MPC from indistinguishability obfuscation. In Yehuda Lindell, editor, *TCC 2014: 11th Theory of Cryptography Conference*, volume 8349 of *Lecture Notes in Computer Science*, pages 74–94, San Diego, CA, USA, February 24–26, 2014. Springer, Heidelberg, Germany.

[GM82] Shafi Goldwasser and Silvio Micali. Probabilistic encryption and how to play mental poker keeping secret all partial information. In *14th Annual ACM Symposium on Theory of Computing*, pages 365–377, San Francisco, CA, USA, May 5–7, 1982. ACM Press.

[GMM+16] Sanjam Garg, Eric Miles, Pratyay Mukherjee, Amit Sahai, Akshayaram Srinivasan, and Mark Zhandry. Secure obfuscation in a weak multilinear map model. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B: 14th Theory of Cryptography Conference, Part II*, volume 9986 of *Lecture Notes in Computer Science*, pages 241–268, Beijing, China, October 31 – November 3, 2016. Springer, Heidelberg, Germany.

[GP20] Romain Gay and Rafael Pass. Indistinguishability obfuscation from circular security. Cryptology ePrint Archive, Report 2020/1010, 2020.

[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 Annual ACM Symposium on Theory of Computing*, pages 197–206, Victoria, BC, Canada, May 17–20, 2008. ACM Press.

[GR07] Shafi Goldwasser and Guy N. Rothblum. On best-possible obfuscation. In Salil P. Vadhan, editor, *TCC 2007: 4th Theory of Cryptography Conference*, volume 4392 of *Lecture Notes in Computer Science*, pages 194–213, Amsterdam, The Netherlands, February 21–24, 2007. Springer, Heidelberg, Germany.

[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, *Advances in Cryptology – CRYPTO 2013, Part I*, volume 8042 of *Lecture Notes in Computer Science*, pages 75–92, Santa Barbara, CA, USA, August 18–22, 2013. Springer, Heidelberg, Germany.

[Had00] Satoshi Hada. Zero-knowledge and code obfuscation. In Tatsuaki Okamoto, editor, *Advances in Cryptology – ASIACRYPT 2000*, volume 1976 of *Lecture Notes in Computer Science*, pages 443–457, Kyoto, Japan, December 3–7, 2000. Springer, Heidelberg, Germany.

[HJ16] Yupu Hu and Huiwen Jia. Cryptanalysis of GGH map. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology – EUROCRYPT 2016, Part I*, volume 9665 of *Lecture Notes in Computer Science*, pages 537–565, Vienna, Austria, May 8–12, 2016. Springer, Heidelberg, Germany.

[IP07] Yuval Ishai and Anat Paskin. Evaluating branching programs on encrypted data. In Salil P. Vadhan, editor, *TCC 2007: 4th Theory of Cryptography Conference*, volume 4392 of *Lecture Notes in Computer Science*, pages 575–594, Amsterdam, The Netherlands, February 21–24, 2007. Springer, Heidelberg, Germany.

[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, *Advances in Cryptology – EUROCRYPT 2019, Part I*, volume 11476 of *Lecture Notes in Computer Science*, pages 251–281, Darmstadt, Germany, May 19–23, 2019. Springer, Heidelberg, Germany.

[JLS20] Aayush Jain, Huijia Lin, and Amit Sahai. Indistinguishability obfuscation from well-founded assumptions. Cryptology ePrint Archive, Report 2020/1003, 2020.

[Kil88] Joe Kilian. Founding cryptography on oblivious transfer. In *20th Annual ACM Symposium on Theory of Computing*, pages 20–31, Chicago, IL, USA, May 2–4, 1988. ACM Press.

[Lin16] Huijia Lin. Indistinguishability obfuscation from constant-degree graded encoding schemes. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology – EUROCRYPT 2016, Part I*, volume 9665 of *Lecture Notes in Computer Science*, pages 28–57, Vienna, Austria, May 8–12, 2016. Springer, Heidelberg, Germany.

[Lin17] Huijia Lin. Indistinguishability obfuscation from SXDH on 5-linear maps and locality-5 PRGs. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology – CRYPTO 2017, Part I*, volume 10401 of *Lecture Notes in Computer Science*, pages 599–629, Santa Barbara, CA, USA, August 20–24, 2017. Springer, Heidelberg, Germany.

[LPST16] Huijia Lin, Rafael Pass, Karn Seth, and Sidharth Telang. Indistinguishability obfuscation with non-trivial efficiency. In Chen-Mou Cheng, Kai-Min Chung, Giuseppe Persiano, and Bo-Yin Yang, editors, *PKC 2016: 19th International Conference on Theory and Practice of Public Key Cryptography, Part II*, volume 9615 of *Lecture Notes in Computer Science*, pages 447–462, Taipei, Taiwan, March 6–9, 2016. Springer, Heidelberg, Germany.

[LT17] Huijia Lin and Stefano Tessaro. Indistinguishability obfuscation from trilinear maps and block-wise local PRGs. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology – CRYPTO 2017, Part I*, volume 10401 of *Lecture Notes in Computer Science*, pages 630–660, Santa Barbara, CA, USA, August 20–24, 2017. Springer, Heidelberg, Germany.

[LV16] Huijia Lin and Vinod Vaikuntanathan. Indistinguishability obfuscation from DDH-like assumptions on constant-degree graded encodings. In Irit Dinur, editor, *57th Annual Symposium on Foundations of Computer Science*, pages 11–20, New Brunswick, NJ, USA, October 9–11, 2016. IEEE Computer Society Press.

[MSZ16] Eric Miles, Amit Sahai, and Mark Zhandry. Annihilation attacks for multilinear maps: Cryptanalysis of indistinguishability obfuscation over GGH13. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology – CRYPTO 2016, Part II*, volume 9815 of *Lecture Notes in Computer Science*, pages 629–658, Santa Barbara, CA, USA, August 14–18, 2016. Springer, Heidelberg, Germany.

[MZ18] Fermi Ma and Mark Zhandry. The MMap strikes back: Obfuscation and new multilinear maps immune to CLT13 zeroizing attacks. In Amos Beimel and Stefan Dziembowski, editors, *TCC 2018: 16th Theory of Cryptography Conference, Part II*, volume 11240 of *Lecture Notes in Computer Science*, pages 513–543, Panaji, India, November 11–14, 2018. Springer, Heidelberg, Germany.

[OPP14] Rafail Ostrovsky, Anat Paskin-Cherniavsky, and Beni Paskin-Cherniavsky. Maliciously circuit-private FHE. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology – CRYPTO 2014, Part I*, volume 8616 of *Lecture Notes in Computer Science*, pages 536–553, Santa Barbara, CA, USA, August 17–21, 2014. Springer, Heidelberg, Germany.

[PRS17] Chris Peikert, Oded Regev, and Noah Stephens-Davidowitz. Pseudorandomness of ring-LWE for any ring and modulus. In Hamed Hatami, Pierre McKenzie, and Valerie King, editors, *49th Annual ACM Symposium on Theory of Computing*, pages 461–473, Montreal, QC, Canada, June 19–23, 2017. ACM Press.

[PVW08] Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. A framework for efficient and composable oblivious transfer. In David Wagner, editor, *Advances in Cryptology – CRYPTO 2008*, volume 5157 of *Lecture Notes in Computer Science*, pages 554–571, Santa Barbara, CA, USA, August 17–21, 2008. Springer, Heidelberg, Germany.

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *37th Annual ACM Symposium on Theory of Computing*, pages 84–93, Baltimore, MA, USA, May 22–24, 2005. ACM Press.

[SW14] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In David B. Shmoys, editor, *46th Annual ACM Symposium on Theory of Computing*, pages 475–484, New York, NY, USA, May 31 – June 3, 2014. ACM Press.

[WW20] Hoeteck Wee and Daniel Wichs. Candidate obfuscation via oblivious lwe sampling. Cryptology ePrint Archive, Report 2020/1042, 2020.

Indistinguishability Obfuscation from Circular Security

Romain Gay*
IBM Zurich
romain.rgay@gmail.com

Rafael Pass†
Cornell Tech
rafael@cs.cornell.edu

October 26, 2020

Abstract

We show the existence of indistinguishability obfuscators ($i\mathcal{O}$) for general circuits assuming subexponential security of:

(a) the Learning with Error (LWE) assumption (with subexponential modulus-to-noise ratio);

(b) a *circular security conjecture* regarding the Gentry-Sahai-Water's (GSW) encryption scheme.

The circular security conjecture states that a notion of leakage-resilient security (that we prove is satisfied by GSW assuming LWE) is retained in the presence of an encryption of the secret key.

Our work thus places $i\mathcal{O}$ on qualitatively similar assumptions as unlevelled FHE, for which known constructions also rely on a circular security conjecture.

*Work done in part while at Cornell Tech

†Supported in part by NSF Award SATC-1704788, NSF Award RI-1703846, AFOSR Award FA9550-18-1-0267, DARPA SIEVE award HR00110C0086, and a JP Morgan Faculty Award. This research is based upon work supported in part by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA), via 2019-19-020700006. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of ODNI, IARPA, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright annotation therein.

1 Introduction

The goal of *program obfuscation* is to "scramble" a computer program, hiding its implementation details (making it hard to "reverse-engineer"), while preserving its functionality (i.e its input/output behavior). In recent years, the notion of *indistinguishability obfuscation ($i\mathcal{O}$)* [BGI+01, GGH+13b] has emerged as the central notion of obfuscation in the cryptographic literature: roughly speaking, this notion requires that obfuscations $i\mathcal{O}(\Pi^1)$, $i\mathcal{O}(\Pi_2)$ of any two *functionally equivalent* circuits Π^1 and Π_2 (i.e. whose outputs agree on all inputs) from some class $\mathcal{C}$ (of circuits of some bounded size) are computationally indistinguishable.

On the one hand, this notion of obfuscation is strong enough for a plethora of amazing applications (see e.g. [SW14, BCP14, BZ14, GGHR14, KNY14, KMN+14, BGL+15, CHJV14, KLW15, CLP15, BPR15, BPW16, BP15]). On the other hand, it may also plausibly exist, whereas stronger notion of obfuscations have run into strong impossibility results, even in idealized models (see e.g. [BGI+01, GK05, CKP15, Ps16, MMN15, LPST16]). Since the breakthrough of Garg, Gentry, Halevi, Raykova, Sahai and Waters [GGH+13b] that presented the first $i\mathcal{O}$ candidate, there has been an intensive effort toward obtaining a construction of $i\mathcal{O}$ based on some form of well-studied/nice assumptions. The original work [GGH+13b] provided a *candidate* construction based on high-degree multilinear maps (MLMs) [GGH13a, CLT13, GGH15, CLT15]; there was no proof of security based on an intractability assumption. [PST14] provided the first construction with a reduction-based proof of security, based on a strong notion of security for MLMs, similar to a sort of "Uber assumption". [GLSW14] provided a construction based on a more concrete assumption relying on composite-order MLMs. Unfortunately, both assumptions have been broken for specific candidate constructions of MLMs [CHL+15, MF15].

$i\mathcal{O}$ from FE or $Xi\mathcal{O}$. Subsequently, several works have been constructing $i\mathcal{O}$ from seemingly weaker primitives, such as Functional Encryption (FE) [AJ15, BV15] or $Xi\mathcal{O}$ [LPST16], while only using standard assumptions, such as Learning with Error (LWE)[1]. For both constructions, we actually need to rely on *subexponentially-secure* constructions of either FE or $Xi\mathcal{O}$, as well as subexponential security of LWE. Let us recall the notion of $Xi\mathcal{O}$ as it will be useful to us: roughly speaking, an $Xi\mathcal{O}$ is an $i\mathcal{O}$ with a very weak "exponential" efficiency requirement: the obfuscator is allowed to run in polynomial time in the size of the truth table of the function to be obfuscated, and it is only required that its outputs a program that "slightly" compresses the truth table (technically, it is sublinear in its size).

A breakthrough result by Lin [Lin16] showed how to obtain $i\mathcal{O}$ from *constant-degree* MLMs (plus standard assumptions), overcoming the black-box barriers in [Ps16, MMN15]. Her construction relies on the connection between FE and $i\mathcal{O}$. Following this result, a sequence of works (see e.g. [LV16, Lin17, LT17, AJKS18, JS18, JLMS19, AJL+19, GJLS20]) reduced the assumptions and the degree of the MLM — all the way down to 2-linear maps a.k.a. pairings— relying on certain types of low-degree pseudorandom generators (PRGs) to build FE. This culminated in the work of [GJLS20], whose security rely on the LWE assumption with binary errors in the presence of some PRG leakage, which despite being quite elegant, is new to their work, and as such, has not been significantly crypt-analyzed. Another line of work [Agr19, AP20] replace the use of 2-linear maps used by the aforementioned works by a noisy linear FE for inner products. While being plausibly post-quantum, these constructions are heuristic and do not provide a security reduction to a simple assumption.

A recent work by Brakerski et al [BDGM20a] presents a new type of candidate construction

[1]Note that we are omitting some works that build $i\mathcal{O}$ without going through FE or $Xi\mathcal{O}$, such as [GJK18] that gives a direct heuristic construction of $i\mathcal{O}$ from tensor products, or [BIJ+20] that describes a candidate from Affine Determinant Programs. None of these provide a security proof.

of $Xi\mathcal{O}$ by combining a fully-homomorphic encryption (FHE) and a linear-hommorphic encryption (LHE) with certain nice properties (which can be instantiated by the Damgård-Jurik (DJ) [DJ01] encryption scheme whose security relies on the Decisional Composite Residuosity (DCR) assumption), and relying on a random oracle. More precisely, they define a new primitive called "split-FHE" and provide a candidate construction of it based on the above primitives and a random oracle, and next show how split-FHE implies $Xi\mathcal{O}$ (which by earlier work implies $i\mathcal{O}$ under standard assumptions). We highlight that [BDGM20a] does not provide any proof of security of the split-FHE construction (even in the random oracle model), but rather informally argue some intuitions, which include a) *circular security* (more on this below) of the FHE and the LHE, and b) a *"correlation conjecture"* that the FHE randomness (after FHE evaluations) does not correlate "too much" with the messages being encrypted. The correlation conjecture is not formalized, as the FHE randomness in known construction actually *does* depend on the message, so the authors simply conjecture that this correlation cannot be exploited by an attacker to break security of the $i\mathcal{O}$ (they also provide heuristic methods to weaken the correlations); as such they only get a heuristic construction.

Summarizing the above, while there have been enormous progress on realizing $i\mathcal{O}$, known constructions are either based on assumptions that are not well understood (high-degree MLMs, various low-degree PRGs assumptions and LWE with leakage type of assumptions), or the construction candidates simply do not have proofs of security.

1.1 Our Results

In this work, we provide a new $i\mathcal{O}$ construction assuming subexponential security of (a) the LWE assumption (with subexponential modulus-to-noise ratio), and (b) an (in our eyes) natural *circular security assumption* w.r.t the Gentry-Sahai-Water's (GSW) [GSW13] FHE scheme.

On a high-level, our approach follows that in [BDGM20a], but we show how to remove the heuristic arguments while instead relying on a concrete circular security assumption. We believe this constitutes strong evidence for the existence of $i\mathcal{O}$, and places $i\mathcal{O}$ on a qualitatively similar footing as *unlevelled* FHE (i.e. an FHE that support an a-priori unbounded polynomial number of operations), for which known constructions also rely on a circular security conjecture [Gen09]. We emphasize that the type of circular security conjecture that we rely on is stronger and more complex than the "plain" circular security conjecture used for unlevelled FHE. Yet on a philosophical level, we do not see any concrete evidence for why the plain circular security is more believable.[2]

Circular security. Circular security of encryption schemes [CL01, BRS02] considers a scenario where the attacker gets to see not only encryptions of messages, but also *encrypted key cycles*. The simplest form of circular security, referred to as *1-circular security*, requires that security holds even if the attacker gets to see not only the public key pk and an encryption $\mathsf{Enc}_{\mathsf{pk}}(\mathbf{m})$ of a message $\mathbf{m}$ (to be secured), but also an encryption $\mathsf{Enc}_{\mathsf{pk}}(\mathsf{sk})$ of the secret key sk. That is, we require that for any two messages $\mathbf{m}^0, \mathbf{m}^1$, $\mathsf{Enc}_{\mathsf{pk}}(\mathsf{sk}||\mathbf{m}^0)$ is indistinguishable from $\mathsf{Enc}_{\mathsf{pk}}(\mathsf{sk}||\mathbf{m}^0)$. A slightly more complex type of circular security, referred to as *2-circular security*, considers an encrypted key cycle of size 2 where the attacker gets to see $\mathsf{Enc}^1_{\mathsf{pk}_1}(\mathsf{sk}_2), \mathsf{Enc}^2_{\mathsf{pk}_2}(\mathsf{sk}_1)$ w.r.t. two (potentially different) encryption schemes $\mathsf{Enc}^1, \mathsf{Enc}^2$. Encrypted key cycles commonly arise in applications of encryption scheme such as storage systems (e.g. BitLocker disk encryption utility), anonymous credentials [CL01] and most recently to construct (unlevelled) FHE [Gen09].

We refer to the assumption that:

If Enc *is secure, then 1-circular security holds for* Enc.

[2] See Section 1.4 for an extended comparison.

as the *1-circular security conjecture (*$\mathsf{1CIRC}$*) w.r.t* Enc. (We may also consider a subexponential version of this conjecture which is identically defined except that "security" is replaced by "subexponential security".) At first sight, one may be tempted to hope that circular security holds w.r.t. *all* secure encryption schemes— after all, the attacker never actually gets to see the secret key, but rather an encryption of it, which intuitively should hide it by semantic security of the encryption schemes. Yet, in recent years, counter examples to both 1-circular and 2-circular security have been found (see e.g. [ABBC10, GH10, CGH12, Rot13, MO14, KRW15, BHW15, KW16, GKW17, WZ17]). However, all known counter examples are highly artificial, and require carefully embedding some trapdoor mechanism in the encryption scheme that enables decrypting the cirphertext once you see an encryption of the secret key. As far as we are aware, no "natural" counterexamples are known. Indeed, a common heuristic consists of simply assuming that 1-circular security holds for all "natural" encryption schemes that are secure; that is, $\mathsf{1CIRC}$ holds for all "natural" encryption schemes —we refer to this as the $\mathsf{1CIRC}$ heuristic. It is similar to the Random Oracle Heuristic [BR93]: while "contrived" counterexamples are known (see e.g., [CGH98, MRH04]), it is still commonly used for the design of practical protocols.

Leakage-resilient Circular Security. In this work, we rely on the assumption that stronger forms of security are preserved in the presence of a key cycle. More precisely, we consider a notion of $\mathcal{O}$-leakage resilient security where $\mathcal{O}$ is some particular *randomness leakage oracle*; this notion enhances the standard semantic security notion by providing the attacker with access to an oracle $\mathcal{O}(\mathbf{m}, \mathbf{r})$ that is parametrized by the message $\mathbf{m}$ being encrypted and the randomness $\mathbf{r}$ under which it is encrypted, while restricting the attacker to making only "valid" leakage queries (that do not trivially leak information about the message—this is formalized by letting the oracle output $\perp$ whenever a query is invalid, and saying that the attacker fails whenever this happens). We next define the notion of *1-circular $\mathcal{O}$-leakage resilient security* analogously to 1-circular security, and also define a $\mathsf{1CIRC}^{\mathcal{O}}$ conjecture (resp. a subexponential $\mathsf{1CIRC}^{\mathcal{O}}$ conjecture) in the same way as the $\mathsf{1CIRC}$ conjecture except that "security" is replaced by "$\mathcal{O}$-leakage resilient security"; that is, $\mathsf{1CIRC}^{\mathcal{O}}$ holds w.r.t an encryption scheme Enc if the following holds:

If Enc *is $\mathcal{O}$-leakage resilient secure, then 1-circular $\mathcal{O}$-leakage resilient security holds.*

Note that we cannot hope that $\mathsf{1CIRC}^{\mathcal{O}}$ security holds for *all* oracles $\mathcal{O}$, even with respect to "natural" encryption schemes: simply consider an oracle $\mathcal{O}(\mathbf{m}, \mathbf{r})$ that outputs the message $\mathbf{m}$ iff $\mathbf{m}$ is a valid secret key (just as in the counterexample to "plain" 1-circular security for string encryption). Thus, for $\mathsf{1CIRC}^{\mathcal{O}}$ to be meaningful, we need to restrict not only to "natural" encryption schemes, but also to "natural" oracles $\mathcal{O}$.

Our main theorem shows that for a natural leakage oracle $\mathcal{O}_{\mathsf{SRL}}$—which will be referred to as the "shielded randomness leakage (SRL) oracle"—$\mathsf{1CIRC}^{\mathcal{O}_{\mathsf{SRL}}}$ w.r.t. the GSW encryption scheme together with the LWE assumption implies the existence of $i\mathcal{O}$.

Theorem 1.1 (Informally stated)**.** *Assume the subexponential security of the LWE assumption (with subexponential modulus-to-noise ratio) and the subexponential* $\mathsf{1CIRC}^{\mathcal{O}_{\mathsf{SRL}}}$ *conjecture w.r.t. GSW. Then, $i\mathcal{O}$ exists for the class of polynomial-size circuits.*

In the sequel, we refer to $\mathcal{O}_{\mathsf{SRL}}$-leakage resilient security (resp. 1-circular $\mathcal{O}_{\mathsf{SRL}}$-leakage resilient security) as SRL-security (resp 1-circular SRL security). We highlight that whereas our main theorem only relies on the notion of 1-circular SRL security, a notion of 2-circular SRL security (which is analogously defined) will be instrumental towards proving our final result. We proceed to explain the notion of SRL security and how the above theorem is proven.

1.2 Shielded Randomness Leakage (SRL) Security

As mentionned above, we consider a notion of *shielded randomness leakage (SRL)* security for FHE. Roughly speaking, given two messages $\mathbf{m}^0, \mathbf{m}^1$, the attacker gets to see an FHE encryption $\mathbf{c} = \mathsf{FHE}(\mathbf{m}^b; \mathbf{r})$ of $\mathbf{m}^b$ for a randomly selected $b \in \{0,1\}$, and next gets access to a "leakage oracle" $\mathcal{O}_{\mathsf{SRL}}(\mathbf{m}^b, \mathbf{r})$ which upon every invocation sends the attacker an "extra noisy" encryption $\mathbf{c}^\star = \mathsf{FHE}(0; \mathbf{r}^\star)$ of 0—we will refer to the random string $\mathbf{r}^\star$ as the "shield". Next, the attacker can select some functions f and values α such that $f(\mathbf{m}^b) = \alpha$—that is, we restrict the attacker to picking functions for which it *knows the output* when applying the function to the message $\mathbf{m}^b$; if $f(\mathbf{m}^b) \neq \alpha$, the attacker directly fails in the game. (The reason why we add this restriction on the attacker will soon become clear). Finally, the oracle homomorphically evaluates f on the ciphertext $\mathbf{c}$, letting $\mathbf{c}_f = \mathsf{FHE}(f(\mathbf{m}); \mathbf{r}_f)$ denote the evaluated ciphertext, and returns $\mathbf{r}^\star - \mathbf{r}_f$. That is, the attacker gets back the randomness $\mathbf{r}_f$ of the evaluated ciphertext masked by the "shield" $\mathbf{r}^\star$, and as usual, the attacker's goal is to guess the bit b. The reason why the attacker is restricted to picking functions f for which it knows the output α is that for the FHE we consider, given $\mathbf{c}^\star$ and $\mathbf{c}_f$, the attacker can compute $\mathbf{c}^\star - \mathbf{c}_f = \mathsf{FHE}(0 - f(\mathbf{m}^b); \mathbf{r}^\star - \mathbf{r}_f)$ and thus knowing $\mathbf{r}^\star - \mathbf{r}_f$ reveals $f(\mathbf{m}^b)$. So, by restricting to attackers that already know $\alpha = f(\mathbf{m}^b)$, intuitively, $\mathbf{r}^\star - \mathbf{r}_f$ does not reveal anything else. Indeed, we formally prove that under the LWE assumption, the GSW encryption scheme is SRL-secure (i.e. $\mathcal{O}_{\mathsf{SRL}}$-leakage resilient secure).

Theorem 1.2 (Informally stated)**.** *Assume the LWE assumption holds (with subexponential modulus-to-noise ratio). Then, the GSW scheme is SRL-secure.*

On a very high-level, the idea behind the proof is that the encryption $\mathbf{c}^\star$ is a projection, $h_{\mathbf{A}}(\mathbf{r}^\star) = \mathbf{A}\mathbf{r}^\star \in \mathbb{Z}_N$, where the randomness $\mathbf{r}^\star$ used to produce $\mathbf{c}^\star$ is a vector in $\mathbb{Z}_N^\ell$ and $\mathbf{A}$ is a matrix in $\mathbb{Z}_N^{n \times \ell}$ where $\ell \gg n$, that is, the map $h_{\mathbf{A}}$ that describes the encryption is compressing. Therefore, some "components" of the "shield" $\mathbf{r}^\star$ remain information-theoretically hidden. And this enables hiding the same components of $\mathbf{r}_f$; furthemore, the components that are not hidden by $\mathbf{r}^\star$ are actually already revealed by $f(\mathbf{m}^b)$, which the attacker knows (as we require it to output $\alpha = f(\mathbf{m}^b)$). The formal proof of this proceeds by considering a (simplified) variant of the Micciancio-Peikert lattice trapdoor method [MP12] for generating the matrix $\mathbf{A}$ (which is part of the public key for GSW) together with a trapdoor that enables sampling short preimages of $h_{\mathbf{A}}$ (i.e. solving the ISIS problem). Whereas traditional trapdoor preimage sampling methods require the preimage to be sampled according to some specific distribution (typically discrete Gaussian) over preimages, we will consider a somewhat different notion: we require that given a target vector $\mathbf{t}$, the distribution of randomly sampled preimages of $\mathbf{t}$ is statistically close to the distribution obtained by starting with any "short" preimage $\mathbf{w}$ of $\mathbf{t}$ and next adding a randomly sampled preimage of 0. Our proof relies on the fact that randomly sampled preimages can be sufficiently larger than $\mathbf{w}$ to ensure that they "smudge" $\mathbf{w}$—we here rely on the fact that modulus-to-noise ratio is subexponential (which we need anyway for the security of our construction) to enable the smudging[3].

1-Circular SRL security As mentioned, we define 1-circular security as 1-circular $\mathcal{O}_{\mathsf{SRL}}$-leakage resilient security; we emphasize that this security game is identically defined to the "plain" SRL security game (described above), with the only exception being that the challenge message now has the form $\mathsf{sk}||\mathbf{m}^b$ (as opposed to just being $\mathbf{m}^b$).

[3] Another consequence of using smudging is that our lattice trapdoor mechanism and its proof become simpler than [MP12], which uses a polynomial-size modulus instead, for a better efficiency.

1.3 Overview of the $Xi\mathcal{O}$ Construction

We present a construction that makes a modular use of any LHE satisfying certain properties, and whose security relies the 2-circular SRL-security w.r.t. GSW and the LHE (i.e., that SRL security of GSW holds in the presence of a encrypted key cycles of length 2 using GSW and the LHE). To obtain a *subexponentially-secure* $Xi\mathcal{O}$ (which is required to obtain $i\mathcal{O}$ by [LPST16]), we need to strengthen the assumptions to also require subexponential security. Next, we note that the DJ LHE satisfies the desired properties. We prove that a packed version of Regev's encryption scheme [Reg05] that is similar to, but actually different from, the packed construction from [PVW08] does so as well. We refer to our LHE simply as Packed Regev LHE. Finally, we show that 1-circular SRL security of GSW implies 2-circular SRL-security of GSW and Packed Regev[4]. Intuitively, this follows from fact the Packed Regev is provably circularly secure (for the same reasons as it holds for Regev's encryption scheme, as is well known).[5] Taken together, this will allow us to prove Theorem 1.1.

Let us start with the construction assuming 2-circular SRL-security w.r.t. GSW and any LHE satisfying the desired properties. As mentioned, on a high-level, our construction follows similar intuitions as the BDGM construction. We combine an FHE (in our case the GSW FHE) with a (special-purpose) LHE to implement an $Xi\mathcal{O}$. In fact, in our approach, we do not directly construct an $Xi\mathcal{O}$, but rather construct an $Xi\mathcal{O}$ with *preprocessing*—this notion, which relaxes $Xi\mathcal{O}$ by allowing the obfuscator to have access to some *long* public parameter pp, was actually already considered in [LPST16] and it was noted there that subexponentially-secure $Xi\mathcal{O}$ with preprocessing also suffices to get $i\mathcal{O}$.

Towards explaining our approach, let us first recall the approach of BDGM — which relies on the DJ LHE — using a somewhat different language that will be useful for us.

The BDGM construction. The high-level idea is quite simple and very elegant. Recall that an $Xi\mathcal{O}$ is only required to work for programs Π with polynomially many inputs $n = \mathsf{poly}(\lambda)$ where λ is the security parameter, and the obfuscators running time is allowed to be polynomial in n; the only restriction is that the obfuscated code should be sublinear in n—we require a "slight" compression of the truth table. More precisely, the obfuscator is allowed to run in time $\mathsf{poly}(n, \lambda)$ (i.e. polynomial time in the size of the truth table), but must output a circuit of size $\mathsf{poly}(\lambda)n^{1-\varepsilon}$ where $\varepsilon > 0$. Assume that we have access to a special "batched" FHE which enables encrypting (and computing on) long messages of length, say m using a *short randomness* of length $\mathsf{poly}(\lambda)\log(m)$; and furthermore that 1) given the secret key and a ciphertext $\mathbf{c}$, we can efficiently recover the ciphertext randomness 2) given a ciphertext $\mathbf{c}$ and its randomness—which will also be referred to as a "hint"—one can efficiently decrypt. Given such a special FHE, it is easy to construct an $Xi\mathcal{O}$: simply cut the truth table into "chunks" of length n^{ε}, FHE encrypt the program Π, then, homomorphically evalute circuits C_i for indices $i \in [n^{1-\varepsilon}]$ such that given the program Π as input, C_i outputs the i'th "chunk" of the truth table, which we denote by Π_i; finally, release the randomness $\mathbf{r}_i$ (i.e. the "hint") of the evaluated ciphertexts. These hints enable compressing n^{ε} bits into $\mathsf{poly}(\lambda)\log(n^{\varepsilon})$ bits and thus the $Xi\mathcal{O}$ is compressing.[6]

[4]Formally, we need to slightly tweak the Packed Regev scheme to prove this; in particular we need the GSW and Packed Regev scheme to use the same secret key.

[5]Also note that, intuitively, 2-circular (SRL) security w.r.t. GSW and DJ (or Packed Regev) implies 1-circular (SRL) security w.r.t. GSW since given an encrypted 2-cycle, we can always recover a GSW encryption of a GSW secret key sk by decrypting the LHE ciphertext using homomorphic operations. This implication is not quite true as using homomorphic operations, one does not obtain a "fresh" encryption of sk, but we can always add a fresh encryption of 0 to get a slightly noisier fresh encryption of sk.

[6]The reason we need to cut the truth table into chunks instead of directly computing the whole output is that the size of the FHE public key and ciphertexts may grow polynomially with the length of the output of the homomorphic evluation, i.e. the "batching capacity". So the obfuscation is only compressing when we have a large number of chunks.

Unfortunately, none of the known FHE constructions have short randomness. BDGM, however, observes that there are *linear* homomorphic encryptions schemes (LHE), notably the DJ LHE, that satisfy the above requirements. Moreover, many FHEs are batcheable (with "long" randomness) and have "essentially" linear decryption: decryption is an inner product of the ciphertext with the secret key, then rounding. That is, the linear operations yield the plaintext with some additional small decryption noises, that are removing when rounding. So if we start off with such an FHE and additionally release an LHE encryption of the FHE secret key, we can get an FHE with the desired "batcheable with short randomness" requirement: we first homomorphically evaluate the inner product of the FHE ciphertext with the encrypted FHE secret key, then simply release the randomness for the evaluated LHE ciphertext (which now is short).

But there are problems with this approach: (1) since FHE decryption requires performing both a linear operation and *rounding*, we are leaking not only Π_i but also the decryption noises, which is detrimental for the security of the FHE (2) the LHE randomness may actually leak more than just the decrypted LHE plaintext (i.e. something about how the LHE ciphertext was obtained). As BDGM shows, both of these problems can be easily overcome if we have access to many fresh LHE encryptions of some "smudging" noise (which is large enough to smudge the FHE decryption noises)[7]. Therefore, the only remaining problem is to generate these LHE encryptions of smudging noises. This is where the construction in BDGM becomes heuristic: (1) they propose to use a random oracle to generate a long sequence of randomness (2) this sequence of randomness can be interpreted as a sequence of LHE encryptions of uniformly random strings u_i for $i = 1, \ldots, n^{1-\varepsilon}$, since the DJ LHE has dense ciphertext (3) they additionally provide an FHE encryption of the LHE secret key $\overline{\mathsf{sk}}$ (note that there is now a circular security issue), on which they FHE-homomorphically evaluate a function f_i that decrypts the i'th LHE ciphertext produced by the random oracle, and computes $\mathsf{MSB}(u_i)$, the most significant bits of u_i (4) finally they LHE-evaluate the (partial) decryption of the evaluated FHE ciphertext (which encrypts $\mathsf{MSB}(u_i)$); the obtained LHE ciphertext can now be subtracted from the LHE ciphertexts generated by the random oracle, to get an LHE encryption of $u_i - \mathsf{MSB}(u_i)$, which is a noise of the appropriate size, i.e. smudging but not uniform.

One problem with this approach, however, is that while we do obtain an LHE encryption of appropriate smudging noise, it is not actually a fresh ciphertext (with fresh randomness). The issue is that the randomness $\mathbf{r}_{f_i}$ of the evaluated FHE ciphertext of $\mathsf{MSB}(u_i)$ may (and actually will) depend on the randomness of the original LHE ciphertext obtained by the RO. Another problem is that LHE can only compute the first step of an FHE decryption (namely, the linear operations), the LHE encryption obtained actually encrypts a message of the form: $u_i - \mathsf{MSB}(u_i) + \mathsf{noise}_i$. As we know, revealing the extra noise is detrimental for security (this is why we are generating LHE encryptions of smudging noises in the first place). Unfortunately, the extra noise that results from partially decrypting the FHE ciphertext depends on u_i, so the lower-order bits of the latter cannot smudge the former. BDGM here simply assumes that the attacker cannot exploit these correlations, and thus only obtain a heuristic construction.

We shall now see how to obtain the appropriate LHE encryption of smudging noises in a provably secure way, relying on *2-circular SRL-security* of GSW and DJ—that is, $\mathcal{O}_{\mathsf{SRL}}$-leakage resilient circular security of GSW and DJ.

Removing the RO. Our first task will be to remove the use of the RO. That will actually be very easy: as we have already observed, it suffices to get an $Xi\mathcal{O}$ with preprocessing to obtain $i\mathcal{O}$, so instead of using a random oracle, we will simply use a long random string as a public parameter, and interpret it as LHE encryptions of random strings.

[7] They formally prove the security of their scheme in an idealized model with access to an oracle that generates fresh LHE encryptions of smudging noise.

Re-encrypting the FHE. The trickier problem will be to deal with the issue of correlations. We will here rely on the fact that we are considering a particular instantiation of the FHE: namely, using (a batched version of) the GSW encryption scheme. On a high-level, the idea for breaking the correlation is to "refresh" or re-encrypt the evaluated FHE ciphertext (which encrypts $\mathsf{MSB}(u_i)$) to ensure that the randomness is fresh and independent of the evaluations. This way, the decryption noise itself is independent of the evaluated circuit. GSW ciphertexts can be re-randomized simply by adding a fresh extra noisy FHE encryption of 0. How do we get such enecryptions? GSW ciphertexts are not dense, so we cannot put them in the public parameters, and even if they were, we still wouldn't be able to get an encryption of 0 (we would have an encryption of a uniformly random plaintext). The public key of the GSW encryption scheme actually contains a bunch of encryptions of 0, but fewer than the amount we need (or else we wouldn't get a compressing XiO). Instead, we use the public key of the GSW encryption to generate extra noisy encryptions of 0, and we include the (many) random coins $(\mathbf{r}_i^\star)_{i\in[n^{1-\varepsilon}]}$ used to generate these ciphertexts as part of the public parameters of the XiO (recall that the public parameters can be as long as we want). This method does indeed enable us to get a fresh FHE encryption of the most significant bits, and thus the correlation has be broken and intuitively, we should be able to get a provably secure construction. But two obstacles remain: (1) we are revealing the randomness used to re-randomize the ciphertexts, and this could hurt security, or render the re-randomization useless and (2) we still have a circular security issue (as we FHE-encrypt the LHE secret key, and LHE-encrypt the FHE secret key). Roughly speaking, the first issue will be solved by relying on SRL-security of GSW, and the second issue will be solved by our circular security conjecture.

In more detail, we note that the re-randomized evaluated FHE ciphertext of $\mathsf{MSB}(u_i)$ and the public parameters $\mathbf{r}_i^\star$ are statistically close to freshly generated extra noisy FHE encryption of $\mathsf{MSB}(u_i)$ using randomness $\mathbf{r}_i^\star$, and setting the public parameter to $\mathbf{r}_i^\star - \mathbf{r}_{f_i}$, where $\mathbf{r}_{f_i}$ is the randomness of the evaluated ciphertext, before re-randomization. In other words, the re-randomization achieves a notion which we refer to as "weak circuit privacy", where the re-randomized ciphertext is independent of the evaluated function f_i. Furthermore, noisy GSW encryptions of $\mathsf{MSB}(u_i)$ essentially have the form of a noisy GSW encryption of 0, to which $\mathsf{MSB}(u_i)$ is added. So, other than $\mathsf{MSB}(u_i)$, which is truly random, $\mathbf{r}_i^\star - \mathbf{r}_{f_i}$ is simply an SRL leakage on a GSW encryption of the LHE secret key $\overline{\mathsf{sk}}$! Thus, intuitively, security should now follow from circular SRL security of GSW and the LHE.

The final construction. We summarize our final XiO construction with preprocessing. The public parameter pp is a long *random* string that consists of two parts:

- The first part $\mathsf{FHE.PubCoin}$ will be interpreted as a sequence of rerandomization vectors $\mathbf{r}^\star$;
- The second part $\mathsf{LHE.PubCoin}$ will be interpreted as a sequence of LHE encryptions

The obfuscator, given a security parameter λ and a circuit $\Pi : \{0,1\}^{\log n} \to \{0,1\}$, where $n = \mathsf{poly}(\lambda)$ proceeds as follows:

- **Output the public keys of the FHE and LHE:** The obfuscator generates a fresh key-pair $(\overline{\mathsf{pk}}, \overline{\mathsf{sk}})$ for the LHE, and next generate a key-pair $(\mathsf{pk}, \mathsf{sk})$ for the GSW FHE. (To make it easier for the reader to remember which key refers to which encryption scheme, we place a *line* over all keys, ciphertexts and algorithms, that correspond to the *linear* homomorphic encryption.) The modulus N of the GSW encryption is set to be the same that the modulus that defines the message space $\mathbb{Z}_N$ of the LHE scheme. Additionally, it chooses N large enough to enable encrypting messages of size n^ε. Finally, it outputs the public keys $(\mathsf{pk}, \overline{\mathsf{pk}})$.
- **Output an FHE encryption of the circuit:** It outputs an FHE encryption (w.r.t. pk) of the program Π, which we denote by ct_1.

- **Output encrypted key cycle:** It computes ct_2, an FHE encryption of $\overline{\mathsf{sk}}$, and $\overline{\mathsf{ct}}$, an LHE encrytion of sk. It outputs the key cycle $\mathsf{ct}_2, \overline{\mathsf{ct}}$.
- **Output hints:** For every $i \in [n^{1-\varepsilon}]$, it outputs a short "hint" $\mathbf{r}_i$ computed as follows:
  - **Evaluate the circuit:** Homomorphically evaluate the circuit C_i on ct_1 and let ct_i denote the resulting evaluated FHE ciphertext — recall that ct_1 encrypts a program Π, and the circuit C_i takes a input a program Π and outputs the i'th chunk of its truth table.
  - **Compute an FHE encryption $\mathsf{ct}_{\mathsf{MSB},i}$ of $\mathsf{MSB}(u_i)$:** Consider the function $f_i(\Pi, \overline{\mathsf{sk}})$ that ignores the input Π but uses the input $\overline{\mathsf{sk}}$ to decrypt the i'th LHE ciphertext from $\mathsf{LHE.PubCoin}$ into a plaintext u_i and outputs $\mathsf{MSB}(u_i)$. The obfuscator homomorphically evaluates f_i on the ciphertexts $\mathsf{ct}_1, \mathsf{ct}_2$ (where, recall, ct_2 is an encryption of $\overline{\mathsf{sk}}$). Let $\mathsf{ct}_{\mathsf{MSB},i} = \mathsf{FHE}(\mathsf{MSB}(u_i); \mathbf{r}_{f_i})$ denote the resulting evaluated FHE ciphertext.
  - **Rerandomize $\mathsf{ct}_{\mathsf{MSB},i}$ into $\mathsf{ct}'_{\mathsf{MSB},i}$:** It uses the i'th chunk of $\mathsf{FHE.PubCoin}$ to get the randomness $\mathbf{r}^\star_i$; generates an extra noisy FHE encryption of 0 using $\mathbf{r}^\star_i$ and homomorphically adds it to $\mathsf{ct}_{\mathsf{MSB},i}$. Let $\mathsf{ct}'_{\mathsf{MSB},i} = \mathsf{FHE}(\mathsf{MSB}(u_i); \mathbf{r}^\star_i + \mathbf{r}_{f_i})$ denote the new (re-randomized) ciphertext.
  - **Proxy re-encrypt ct_i as an LHE ciphertext $\overline{\mathsf{ct}}_i$:** It uses $\overline{\mathsf{ct}}$ (which, recall, is an LHE encryption of sk) to homomorphically compute the *linear* part of the FHE decryption of ct_i, which yields an LHE encryption of the value $2^\omega \cdot \Pi_i + \mathsf{noise}_i$ where noise_i is an FHE decryption noise, and 2^ω is taken large enough so that the plaintext Π_i can be recovered by rounding.

    Similarly, it homomorphically computes the partial FHE decryption of $\mathsf{ct}'_{\mathsf{MSB},i}$, which yields an LHE encryption of the value $2^{\omega'} \cdot \mathsf{MSB}(u_i) + \mathsf{noise}_{\mathsf{MSB},i}$, where once again $\mathsf{noise}_{\mathsf{MSB},i}$ denotes an FHE decryption noise, and $2^{\omega'} = 1$ for reasons that will become clear later. We rely on the fact that GSW FHE (and many other FHE schemes) admits a flexible "scaled" evaluation algorithm, that can choose which integer 2^ω to use when performing the homomorphic evaluation (this was used also in prior works, including [BDGM20a]). The resulting LHE ciphertext is subtracted from $\mathsf{LHE}(2^\omega \cdot \Pi_i + \mathsf{noise}_i)$, and therefore yields $\mathsf{LHE}(2^\omega \cdot \Pi_i + \mathsf{noise}_i - \mathsf{MSB}(u_i) - \mathsf{noise}_{\mathsf{MSB},i})$.

    Finally, it homomorphically adds the LHE encryption of u_i that is part of the LHE public coins, to obtain $\overline{\mathsf{ct}}_i = \mathsf{LHE}(m_i)$, where $m_i = 2^\omega \cdot \Pi_i + \mathsf{noise}_i - \mathsf{MSB}(u_i) - \mathsf{noise}_{\mathsf{MSB},i} + u_i = 2^\omega \cdot \Pi_i + \mathsf{noise}_i + \mathsf{noise}_{\mathsf{MSB},i} + \mathsf{LSB}(u_i)$, where $\mathsf{LSB}(u_i)$ denotes the least significant bits of u_i.

    The integer ω' is chosen to be equal to 0 so that the smudging noise $\mathsf{LSB}(u_i)$ is directly added to the FHE noises $\mathsf{noise}_i - \mathsf{noise}_{\mathsf{MSB},i}$. As opposed to the value Π_i that we place in the higher-order bits of the plaintext, we need the smudging noise to be at the same level than the FHE noises, so they "blend" together.
  - **Release hint $\mathbf{r}_i$ for LHE ciphertext $\overline{\mathsf{ct}}_i$:** It uses $\overline{\mathsf{sk}}$ to recover the randomness $\mathbf{r}_i$ of $\overline{\mathsf{ct}}_i$ (recall that the LHE we use has a randomness recoverability property), and outputs $\mathbf{r}_i$.

To evaluate the obfuscated program on an input $\mathbf{x} \in \{0,1\}^n$, that pertains to the i'th chunk of the truth table of Π for some $i \in [n^{1-\varepsilon}]$, we compute $\overline{\mathsf{ct}}_i$ just like the obfuscator did (note that this does not require knowing the secret key, but only information contained in the obfuscated code). Finally, we decrypt $\overline{\mathsf{ct}}_i$ using the hint $\mathbf{r}_i$ to recover the message m_i described above (recall that the LHE we use has the property that ciphertexts can be decrypted if you know the randomness). Finally, perform the rounding step of FHE decryption on m_i to obtain Π_i, which contains $\Pi(\mathbf{x})$.

Outline of the security proof. We provide a very brief outline of the security proof. We will rely on the fact that LHE cirphertexts (of random messages) are dense (in the set of bit strings), and additionally on the fact that both the LHE and the FHE we rely on (i.e. DJ and GSW) satisfy what we refer to as a *weak circuit privacy* notion. This notion, roughly speaking, says that *any* encryption of a message x can be rerandomized into fresh (perhaps extra noisy) encryption of $x + y$, by adding a fresh (perhaps extra noisy) encryption of y.

As usual, the proof proceeds via a hybrid argument. We start from an $Xi\mathcal{O}$ obfuscation of a program Π^0 and transition until we get an $Xi\mathcal{O}$ obfuscation of Π^1, where Π^0 and Π^1 are two functionally equivalent circuits of the same size.

- **Hybrid 0: Honest $Xi\mathcal{O}(\Pi^0)$.** The first hybrid is just the honest obfuscation of the circuit Π^0.

- **Hybrid 1: Switch to freshly encrypted $\mathsf{ct}'_{\mathsf{MSB},i}$.** Hybrid 1 proceeds exactly as Hybrid 0 up until the point that the ciphertexts $\mathsf{ct}_{\mathsf{MSB},i}$ get re-encrypted into $\mathsf{ct}'_{\mathsf{MSB},i}$, with the exception that $\mathsf{FHE.PubCoin}$ are not sampled yet. Next, instead of performing the re-encryption, we sample $\mathsf{ct}'_{\mathsf{MSB},i}$ as a *fresh* extra noisy encryption of $\mathsf{MSB}(u_i)$ using randomness $\mathbf{r}^\star_i$, and setting $\mathsf{FHE.PubCoin}$ to be $\mathbf{r}^\star_i - \mathbf{r}_{f_i}$ (recall that $\mathbf{r}_{f_i}$ is the randomness obtained when homomorphically evaluating f_i on the FHE encryption of $\overline{\mathsf{sk}}$). We finally continue the experiment in exactly the same way as in Hybrid 0.

  It follows from the "weak circuit privacy" property of the FHE that Hybrid 0 and Hybrid 1 are statistically close. Note that in Hybrid 1, for each $i \in [n^{1-\varepsilon}]$, the i'th chunk of $\mathsf{FHE.PubCoin}$ can be thought of as SRL leakage on the fresh encryption $\mathsf{ct}'_{\mathsf{MSB},i}$ computed w.r.t. function f_i, which will be useful for us later.

- **Hybrid 2: Switch $\mathsf{LHE.PubCoin}$ to encryptions of random strings.** Hybrid 2 proceeds exactly as Hybrid 1 except that instead of sampling $\mathsf{LHE.PubCoin}$ as a random string, we sample it as fresh LHE encryptions of random strings u_i, for $i = 1, \ldots, n^{1-\varepsilon}$. It follows by the density property of the LHE that Hybrid 2 is statistically close to Hybrid 1.

- **Hybrid 3: Generate $\overline{\mathsf{ct}}_i$ as a fresh encryption.** Hybrid 3 proceeds exactly as Hybrid 2 except that $\overline{\mathsf{ct}}_i$ is generated as a fresh encryption of m_i using fresh randomness $\mathbf{r}_i$, and the i'th chunk of $\mathsf{LHE.PubCoin}$ is instead computed homomorphically by subtracting the LHE encryption of $\mathsf{sk}^\top(\mathsf{ct}_i - \mathsf{ct}_{\mathsf{MSB},i})$ (obtained after homomorphically decrypting ct_i and $\mathsf{ct}'_{\mathsf{MSB},i}$ using $\overline{\mathsf{ct}}$) from the LHE ciphertext $\overline{\mathsf{ct}}_i$. Recall that $m_i = \mathsf{sk}^\top(\mathsf{ct}_i - \mathsf{ct}_{\mathsf{MSB},i}) + u_i$ so the above way of computing the i'th chunk of $\mathsf{LHE.PubCoin}$ ensures that it is valid encryption of u_i as in Hybrid 2, but this time with non-fresh, homomorphically evaluated randomness.

  It follows from the weak circuit privacy property of the LHE that Hybrid 3 and 2 are statistically close.

  Note that it was possible to define this hybrid since $\mathsf{ct}'_{\mathsf{MSB},i}$ remains exactly the same no matter what the $\mathsf{LHE.PubCoin}$ are. This was not true in Hybrid 0, and we introduced Hybrid 1 to break this dependency.

  Note further that in Hybrid 3, we no longer use $\overline{\mathsf{sk}}$ (i.e. the secret key for LHE); previously it was used to recover $\mathbf{r}_i$.

- **Hybrid 4: Generate $\overline{\mathsf{ct}}_i$ without FHE noises.** Hybrid 4 proceeds exactly as Hybrid 3 except that $\overline{\mathsf{ct}}_i$ is generated as a fresh encryption of $m_i = 2^\omega \cdot \Pi^0_i + \mathsf{LSB}(u_i)$, whereas in Hybrid 3, it was generated as fresh encryption of $m_i = 2^\omega \cdot \Pi^0_i + \mathsf{LSB}(u_i) + \mathsf{noise}_i - \mathsf{noise}_{\mathsf{MSB},i}$. That is, we use $\mathsf{LSB}(u_i)$ as a smudging noise to hide the extra noise $\mathsf{noise}_i - \mathsf{noise}_{\mathsf{MSB},i}$. We can do so

since (1) the extra FHE noise is small and independent of $\mathsf{LSB}(u_i)$ (2) the rest of the obfuscated code can be generated from the value $\mathsf{LSB}(u_i) + \mathsf{noise}_i - \mathsf{noise}_{\mathsf{MSB},i}$ only (in particular it does not require to know $\mathsf{LSB}(u_i)$ itself). It follows that Hybrid 4 is statistically close to Hybrid 3.

- **Hybrid 5: Switch to encryption of Π^1:** Hybrid 5 proceeds exactly as Hybrid 4 except that ct_1 is an encryption of Π^1 (instead of Π^0 in prior hybrids).

  Note that other than the encrypted key cycle, we never use the FHE secret key, and due to Hybrid 3, we no longer use the LHE secret key. So, at first sight, Hybrid 5 ought to be indistinguishable from Hybrid 4 by circular security of the FHE and the LHE. Recall that $\mathsf{FHE.PubCoin}$ leaks something about the randomness used by the FHE encryption $\mathsf{ct}'_{\mathsf{MSB},i}$, but the leakage is exactly an SRL leakage (and note that in the experiment we do know the output α_i of the function f_i that is applied to the plaintexts encrypted in $\mathsf{ct}_1, \mathsf{ct}_2$—namely, it is $\mathsf{MSB}(u_i)$ where u_i is a random string selected in the experiment, see Hybrid 2). Thus, indistinguishability of Hybrid 5 and Hybrid 4 follows from 2-circular SRL-security of the FHE and the LHE.

- **Hybrids 6-10:** For $i \in [5]$, Hybrid $5 + i$ is defined exactly as $5 - i$, except that ct_1 be an encryption of Π^1. Statistical closeness of intermediary hybrids follows just as before.

The above sequence of hybrid allows us to conclude the following theorem.

Theorem 1.3 (Informally stated)**.** *Assume the 2-circular SRL-security of the GSW and DJ encryption schemes. Then, there exists an $Xi\mathcal{O}$ for polynomial-size circuits taking inputs of length* $\log(\lambda)$ *where λ is the security parameter.*

An alternative LHE based on Packed Regev. We remark that we can obtain an alternative construction of an LHE with the desired properties by considering an *packed* version of the Regev encryption scheme. Our construction is slightly different, but similar in spirit, to the Packed Regev from [PVW08]. Recall that a (plain) Regev public key consist of a pair $\mathbf{A}, \mathbf{s}^\top\mathbf{A} + \mathbf{e}^\top$, where $\mathbf{A} \leftarrow_{\mathsf{R}} \mathbb{Z}_q^{m\times n}$ with $m \geq n\log(q)$, the vector $\mathbf{s} \leftarrow_{\mathsf{R}} \mathbb{Z}_q^n$ is the secret key, and $\mathbf{e} \in \mathbb{Z}_q^m$ is some small "noise" vector. An encryption of a message μ has the form $\mathbf{Ar}, (\mathbf{s}^\top\mathbf{A}+\mathbf{e}^\top)\mathbf{r}+B\cdot\mu$ where $\mathbf{r} \leftarrow_{\mathsf{R}} \{0,1\}^m$ is the encryption randomness and B is an upper bound on the size of noise (so as to enable decryption). This scheme is linearly homomorphic, but for security, the size of the randomness $|\mathbf{r}|$ needs to be greater than $n\log(q)$, which is more than that size of the message: the randomness is too long for our purposes.

To get succinct decryption hints, we simply reuse the same randomness $\mathbf{r}$ for many encryptions using different secret keys $\mathbf{s}_1, \mathbf{s}_2, \ldots \mathbf{s}_\ell$ and different noises $\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_\ell$. The secret key is now a matrix $\mathbf{S} \in \mathbb{Z}_q^{\ell\times n}$, and the public key becomes $(\mathbf{A}, \mathbf{SA} + \mathbf{E})$ where $\mathbf{E} \in \mathbb{Z}_q^{\ell\times m}$ is a noise matrix. The encryption of a vector of messages $\mu = (\mu_1, \ldots, \mu_\ell)$ is then $(\mathbf{Ar}, (\mathbf{SA} + \mathbf{E})\mathbf{r} + B\mu)$. This is the scheme from [PVW08]. Despite the fact that this encryption is still linearly homomorphic, and has the advantage of having rate-1 ciphertext size, its randomness is not short: to carry on the proof of security, we need to rely on the fact that $\mathbf{r}$ contains enough bits of entropy even when the information $\mathbf{Ar}$ (which is short) and $\mathbf{Er}$ (that is long) is leaked. The can only be true when the dimension of $\mathbf{r}$, m, grows with the number of bits that are batched, ℓ.

Thus, we depart from the scheme in [PVW08] by adding a smudging noise[8] in the ciphertext, to hide the information $\mathbf{Er}$. The ciphertext is of the form: $(\mathbf{Ar}, (\mathbf{SA} + \mathbf{E})\mathbf{r} + \mathbf{e}' + B \cdot \mu)$, where $\mathbf{e}'$

[8] Note that using a carefully crafted noise that needs not be of smudging size, as done in [MP12], we can "unskew" the noise $\mathbf{Er}$ and hide the information of $\mathbf{r}$. We favor clarify of the exposition over efficiency and resort to using smudging noises.

is the extra smudging noise that hides the error term $\mathbf{Er}$, ensuring that we only have the short $\mathbf{Ar}$ leakage and the usual proof can again be applied.

This scheme is still linearly homomorphic, but the encryption randomness is still large, as even though we reuse $\mathbf{r}$, the added noise terms $\mathbf{e}'$ are large. However, we rely on the fact that knowing $\mathbf{e}'$ is not needed for decrypting. Indeed, to decrypt, we just need to know a small vector $\widetilde{\mathbf{r}} \in \mathbb{Z}_q^m$ such that $\mathbf{A}\widetilde{\mathbf{r}} = \mathbf{Ar}$. That can be used to remove the term $\mathbf{SAr}$ from the ciphertext, and recover $B \cdot \nu$ plus some small noise. To sample such vector, we use a standard trapdoor sampling mechanism as in prior works [Ajt96, GPV08, AP09, MP12]. This makes the scheme hintable with succinct hints.

We still have two (minor) obstacles, though. This scheme (as well as Regev's original scheme or the scheme from [PVW08]) does not satisfy two of the other properties needed for our XiO construction: (1) density, and (2) weak circuit privacy. But it almost does. *Extra noisy* ciphertexts, where the noise reaches the bound B are actually dense, and for extra noisy ciphertext, weak circuit privacy also holds (just as it did for GSW). So, we can directly instantiate the LHE in our XiO construction with this Packed Regev construction, as long as we slightly relax the notion of an LHE to just require density when considering extra noisy ciphertexts.

Thus we can conclude:

Theorem 1.4 (Informally stated)**.** *Assume 2-circular SRL-security of the GSW and the Packed Regev encryption schemes holds. Then, there exists an XiO for polynomial-size circuits taking inputs of length $\log(\lambda)$ where λ is the security parameter.*

Relying only on $\mathsf{1CIRC}^{\mathcal{O}_{\mathsf{SRL}}}$ w.r.t. GSW. We finally explain how to base security solely on 1-circular SRL security w.r.t. GSW. We proceed in two steps. First, we remark that in our XiO construction, security still holds if both the FHE and LHE use the same secret key (as long as 2-circular SRL security of the two schemes hold in this setting). Next, we present a slight modification of Packed Regev, called Packed-Regev', where the secret key $\mathbf{s}$ is just a vector like in GSW, which it then expanded into a Packed Regev secret key (which is a matrix) by tensoring with the identify matrix. We finally remark that (as is well known for the Regev scheme), 1-circular security directly holds for Packed Regev'—more precisely, given LWE samples (obtained from the GSW public key), we can simulate a Packed Regev' public key, and a Packed Regev' encryption of the secret key. Thus, "same-key" 2-circular SRL security of GSW and Packed-Regev' is implied just by 1-circular security of GSW! We conclude:

Theorem 1.5 (Informally stated)**.** *Assume 1-circular SRL-security of the GSW encryption scheme. Then, there exists an XiO for polynomial-size circuits taking inputs of length $\log(\lambda)$ where λ is the security parameter.*

The proof of Theorem 1.1 is finally concluded by upgrading Theorems 1.2 and 1.5 to apply also in the subexponential regime, relying on the subexponential $\mathsf{1CIRC}^{\mathcal{O}_{\mathsf{SRL}}}$ conjecture, and finally relying on the transformation from subexponentially-secure XiO with pre-processing (and subexponential LWE) to iO [LPST16].

1.4 Comparing Circular SRL-security to "Plain" Circular Security

Let us make a few remarks on the assumption that GSW satisfies 1-circular SRL-security. Clearly, this assumption is stronger than the assumption that GSW satisfies "plain" 1-circular security—simply consider an attacker that does not request any leakage. Additionally, we wish to highlight a few qualitative differences between "plain" circular security w.r.t. GSW and circular SRL security w.r.t. GSW:

- "Plain" circular security of GSW is a simple non-interactive falsifiable assumption. Circular SRL-security is also a (relatively simple) falsifiable assumption, but the security game is now *interactive*; for the type of SRL security needed for our application, a single "parallel" SRL query suffices and such a notion of SRL security can be specified as a 5-round security game.

  As we explain in more detail in Section 3.2.3, for our application, one could define a *non-interactive* falsifiable variant of SRL security—roughly speaking, where the messages and the leakage-selection algorithm are randomly selected—such that the subexponential hardness of this circular "random-SRL" security notion suffices[9], but in our eyes, this non-interactive security game is less natural than the interactive one (and thus does not add much insight).

- It is also worth noting that for the notion of "plain" circular security, an alternative way of defining circular security is to require indistinguishability of encryptions of the secret key and encryptions of 0; this notion (together with non-circular security) implies circular security the way we have defined it (i.e. indistinguishability of encryptions of two messages in the presence of an encryption of the secret key). However, this implication no longer holds in the context of SRL security (see Section 2.8 for more details). And this is why we are directly defining circular security as indistinguishability of encryptions of messages in the presence of an encrypted secret key.

The above two points indicate that the assumption that GSW is circularly-SRL security is both (a-priori) stronger, and also somewhat different from a qualitiative point of view, that the assumption that GSW is just "plain" circularly secure. Yet, in our eyes, the main justifications for believing that GSW is circularly secure hold true also for circular SRL-security:

1. In both cases (plain and SRL), security holds in a non-circular setting, assuming LWE.

2. In both cases (plain and SRL), the security game being considered captures a simple and natural process (albeit for the case of SRL security, it is more complex).

3. Finally, just as for the notion plain circular security, it does not appear simple to even just come up with any *bit*-encryption scheme (such as GSW) that is SRL secure, but not 1-circular SRL secure.[10]

1.5 Concurrent and Subsequent Work

A concurrent and independent breakthrough result by Jain, Lin and Sahai [JLS20] presents a construction of $i\mathcal{O}$ based on subexponential security of well-founded assumptions: (1) the SXDH assumption on asymmetric bilinear groups, (2) the LWE assumption with subexponential modulus-to-noise ration, (3) a Boolean PRG in NC^0, and (4) an LPN assumption over a *large field* and with a small error rate $\frac{1}{\ell^\delta}$ where $\delta > 0$ and ℓ is the dimension of the LPN secret. Assumptions (1) and (2) have widespread use and are considered standard. (3) has also been well-studied in recent years. (4) is a very natural coding problem, but the range of parameters used in (4) differs from most prior works in the cryptographic literature, a majority of which focus on a less sparse error rate (typically a constant) and/or use the field $\mathbb{F}_2$.

[9] This follows from a union bound as the length of both the messages $\mathbf{m}^0, \mathbf{m}^1$ and the description of the leakage-selection algorithm are "short".

[10] Wichs and Zirdelis [WZ17] show that any public-key bit encryption scheme can modified in a way that preserves security yet violates circular security (using a special form of obfuscation that can be satisfied under LWE). The same method can be used to obtain an SRL-secure encryption scheme (by modifying GSW as in [WZ17]) that is not 1-circular SRL secure.

A concurrent and independent work by Wee and Wichs [WW20] presents a new elegant heuristic instantiation of the BDGM paradigm based only on lattice-based primitives. Similarly to us, their construction proceeds by implementing $Xi\mathcal{O}$ with pre-processing. They also state a new security assumption with a circular security flavor (involving a PRF and LWE samples) under which they can prove the security of their construction: Roughly speaking, their construction proceed by reducing $Xi\mathcal{O}$ with pre-processing to the task of "oblivious LWE sampling", and next they provide a heuristic instantiation of a protocol for performing oblivious LWE sampling. Their security assumption is essentially that their protocol is a secure oblivious LWE sampler. It is worth noting that even though they also rely on the BDGM approach to implement $Xi\mathcal{O}$, they manage to directly construct an FHE with short randomness, relying on a "dual" variant of the GSW encryption scheme, thereby completely removing the use of any LHE (whereas we obtain short decryption hints by combining GSW with our Packed Regev).

The initial version of our paper did not contain the LWE-based instantiation of the LHE using Packed Regev (we just had the DJ-based instantiation). Following up on the initial posting of our paper, but concurrently and independently from our LWE-based construction, a preprint by Brakerski et al [BDGM20b] also provides an LWE-based way to instantiate the LHE within our framework. Differently from our construction, however, they rely on a variant of the "Dual Regev" encryption scheme, whereas we rely on regular Regev. They assume 2-circular SRL security holds w.r.t. GSW and their new encryption scheme, whereas we show that for our instantiation of the LHE, it suffices to assume 1-circular SRL security of GSW.

2 Preliminaries and Definitions

In this section, we recall some standard definitions and results. Additionally, we include a formalization of the circular security assumption that we consider.

Attackers, negligible functions and subexponential security. Below, for simplicity of exposition, we provide definitions for *polynomial security* of all the primitives we consider. As usual, we model attackers as *non-uniform probabilistic polynomial-time algorithms*, denoted *nuPPT*. We say that a function $\mu(\cdot)$ is *negligible* if for every polynomial $p(\cdot)$, there exists some $\lambda_0 \in \mathbb{N}$ such that $\mu(\lambda) \leq \frac{1}{p(\lambda)}$ for all $\lambda > \lambda_0$. The security definitions we consider will require that for every nuPPT A, there exists some negligible function μ such that for all λ, A succeeds in "breaking security" w.r.t. the security parameter λ with probability at most $\mu(\lambda)$. All the definitions that we consider can be extended to consider *subexponential security*; this is done by requiring the existence of a constant $\varepsilon > 0$, such that for every non-uniform (probabilistic) attacker $\mathcal{A}$ with running time $\mathsf{poly}(\lambda) \cdot 2^{\lambda^\varepsilon}$, there exists some negligible function μ such that for all λ, $\mathcal{A}$ succeeds in "breaking security" w.r.t. the security parameter λ with probability at most $\mu(\lambda) \cdot 2^{-\lambda^\varepsilon}$ (as opposed to just $\mu(\lambda)$).

Notations. For all $n, m \in \mathbb{N}$, we write $[-n, m] = \{-n, -n+1, \ldots, m\}$, $[n] = [1, n]$. For all $\alpha, \beta \in \mathbb{R}$ such that $\beta > \alpha$, we denote by $(\alpha, \beta) = \{x \in \mathbb{R}, \alpha < x < \beta\}$. For all $v_1, \ldots, v_n \in \mathbb{Z}$, we denote by $\mathbf{v} = (v_1, \ldots, v_n)$ the column vector in $\mathbb{Z}^n$. For all probabilistic polynomial time (PPT) algorithm $\mathcal{A}$, we denote by $y \leftarrow \mathcal{A}(x)$ the random process of running $\mathcal{A}$ on input x and obtaining the output y. For all sets $\mathcal{S}$, we denote by $x \leftarrow_{\mathsf{R}} \mathcal{S}$ the process of sampling a random element x uniformly over $\mathcal{S}$. For any sequence $x = \{x_\lambda\}_{\lambda \in \mathbb{N}}$, we write $x \in 2^{\mathsf{poly}(\lambda)}$ if there exists a polynomial $p(\cdot)$ such that for all $\lambda \in \mathbb{N}$, $x_\lambda \in \mathbb{N}$ and the bit size of x_λ is less than $p(\lambda)$. For all functions $B(\cdot)$, we say an ensemble $\{\mathcal{D}_\lambda\}_{\lambda \in \mathbb{N}}$ of distributions that output in $\mathbb{Z}$ is B-bounded if for all $\lambda \in \mathbb{N}$, all x in the support of $\mathcal{D}_\lambda$, we have $|x| < B(\lambda$. We say an ensemble $\{\mathcal{D}_\lambda\}_{\lambda \in \mathbb{N}}$ is polynomially-bounded if there exists a polynomials $p(\cdot)$ such that $\{\mathcal{D}_\lambda\}_{\lambda \in \mathbb{N}}$ is p-bounded. We say two ensembles $\{\mathcal{D}^0_\lambda\}_{\lambda \in \mathbb{N}}$ and $\{\mathcal{D}^1_\lambda\}_{\lambda \in \mathbb{N}}$ are statistically

close (without specifying the statistical distance) and we write $\{\mathcal{D}^0_\lambda\}_{\lambda\in\mathbb{N}} \approx_s \{\mathcal{D}^1_\lambda\}_{\lambda\in\mathbb{N}}$ when for all $\lambda \in \mathbb{N}$, the distributions $\mathcal{D}^0_\lambda$ and $\mathcal{D}^1_\lambda$ have statistical distance $2^{-\Omega(\lambda)}$.

2.1 Standard Lemmas

We first recall a special case of the leftover hash lemma from [ILL89].

Lemma 2.1 (leftover hash lemma). *For all $\lambda, q, d, m \in \mathbb{N}$ such that $m \geq d\lceil\log(q)\rceil + 2\lambda$, the statistical distance between the following distributions is upper bounded by $2^{-\lambda}$:*

$$\left\{\mathbf{A} \leftarrow_{\mathsf{R}} \mathbb{Z}_q^{d\times m}, \mathbf{r} \leftarrow_{\mathsf{R}} [-1,1]^m : (\mathbf{A}, \mathbf{Ar})\right\}$$
$$\left\{\mathbf{A} \leftarrow_{\mathsf{R}} \mathbb{Z}_q^{d\times m}, \mathbf{u} \leftarrow_{\mathsf{R}} \mathbb{Z}_q^{d} : (\mathbf{A}, \mathbf{u})\right\}.$$

We will also make use of the following standard "smudging" lemma (see e.g. [AJL+12] for an explicit proof of this lemma).

Lemma 2.2 (Smudging). *For all $B, B' \in \mathbb{N}$ such that $B < B'$, all $x \in [-B, B]$, the statistical distance between the following distributions is upper bounded by $\frac{B}{B'}$:*

$$\left\{u \leftarrow_{\mathsf{R}} [-B', B'] : u\right\}$$
$$\left\{u \leftarrow_{\mathsf{R}} [-B', B'] : u + x\right\}.$$

2.2 Indistinguishability

We start by recalling the standard definition of computational indistinguishability [GM84].

Definition 2.1 (Computational indistinguishability). *Two ensembles $\{\mathcal{D}^0_\lambda\}_{\lambda\in\mathbb{N}}$ and $\{\mathcal{D}^1_\lambda\}_{\lambda\in\mathbb{N}}$ are said to be* computationally indistinguishable *if for every nuPPT $\mathcal{A}$ there exists a negligible function μ such that for every $\lambda \in \mathbb{N}$,*

$$\left|\Pr[\mathcal{A}(1^\lambda, \mathcal{D}^0_\lambda) = 1] - \Pr[A(1^\lambda, \mathcal{D}^1_\lambda) = 1]\right| \leq \mu(\lambda)$$

We write $\{\mathcal{D}^0_\lambda\}_{\lambda\in\mathbb{N}} \approx_c \{\mathcal{D}^1_\lambda\}_{\lambda\in\mathbb{N}}$ to denote that $\{\mathcal{D}^0_\lambda\}_{\lambda\in\mathbb{N}}$ and $\{\mathcal{D}^1_\lambda\}_{\lambda\in\mathbb{N}}$ are computationally indistinguishable.

2.3 Definition of iO

We recall the definition of $i\mathcal{O}$ [BGI+01, GGH+13b]. Given polynomials $n(\cdot), s(\cdot), d(\cdot)$, let $\mathcal{C}_{n,s,d} = \{\mathcal{C}_\lambda\}_{\lambda\in\mathbb{N}}$ denote the class of circuits such that for all $\lambda \in \mathbb{N}$, $\mathcal{C}_\lambda$ is the set of circuits with input size $n(\lambda)$, size at most $s(\lambda)$ and depth at most $d(\lambda)$. We say that a sequence of circuits $\{\Pi_\lambda\}_{\lambda\in\mathbb{N}}$ is contained in $\{\mathcal{C}_\lambda\}_{\lambda\in\mathbb{N}}$ (denoted by $\{\Pi_\lambda\}_{\lambda\in\mathbb{N}} \in \{\mathcal{C}_\lambda\}_{\lambda\in\mathbb{N}}$) if for all $\lambda \in \mathbb{N}$, $\Pi_\lambda \in \mathcal{C}_\lambda$.

Definition 2.2 ($i\mathcal{O}$ for $\mathsf{P/poly}$). *We say that $i\mathcal{O}$ exists for $\mathsf{P/poly}$ if for all polynomials $n(\cdot), s(\cdot), d(\cdot)$, there exists a tuple of PPT algorithms $(\mathsf{Obf}, \mathsf{Eval})$ such that the following holds:*

- **Correctness:** *For all $\{\Pi_\lambda\}_{\lambda\in\mathbb{N}} \in \mathcal{C}_{n,s,d}$, there exists a negligible function μ such that for all $\lambda \in \mathbb{N}$, all $\mathbf{x} \in \{0,1\}^{n(\lambda)}$,*

$$\Pr[\widetilde{\Pi} \leftarrow \mathsf{Obf}(1^\lambda, \Pi_\lambda) : \mathsf{Eval}(1^\lambda, \widetilde{\Pi}, \mathbf{x}) = \Pi(\mathbf{x})] \geq 1 - \mu(n)$$

- **IND-security:** *For all sequences* $\{\Pi_0^\lambda\}_{\lambda\in\mathbb{N}}, \{\Pi_1^\lambda\}_{\lambda\in\mathbb{N}} \in \mathcal{C}_{n,s,d}$ *such that for all* $\lambda \in \mathbb{N}$, Π_0^λ *and* Π_1^λ *are functionally equivalent circuits, the following ensembles are computationally indistinguishable:*

$$\left\{\widetilde{\Pi} \leftarrow \mathsf{Obf}(1^\lambda, \Pi_\lambda^0) : \widetilde{\Pi})\right\}_{\lambda\in\mathbb{N}}$$
$$\left\{\widetilde{\Pi} \leftarrow \mathsf{Obf}(1^\lambda, \Pi_\lambda^1) : \widetilde{\Pi})\right\}_{\lambda\in\mathbb{N}}$$

2.4 Definition of XiO

We recall the definition of $Xi\mathcal{O}$ with pre-processing [LPST16]. We restrict our attention to circuits with input length $O(\log\lambda)$: Given polynomials $n(\cdot), s(\cdot), d(\cdot)$, let $\mathcal{C}_{\log(n),s,d} = \{\mathcal{C}_\lambda\}_{\lambda\in\mathbb{N}}$ denote the class of circuits such that for all $\lambda \in \mathbb{N}$, $\mathcal{C}_\lambda$ is the set of circuits with input size $\log(n(\lambda))$, size at most $s(\lambda)$ and depth at most $d(\lambda)$.

Definition 2.3 ($Xi\mathcal{O}$ for $\mathsf{P}^{\log}/\mathsf{poly}$)**.** *We say* $Xi\mathcal{O}$ *exists for* $\mathsf{P}^{\log}/\mathsf{poly}$ *if there exists a polynomial* $p(\cdot)$ *and a constant* $\varepsilon \in (0,1)$*, such that for all polynomials* $n(\cdot), s(\cdot), d(\cdot)$*, there exists a tuple of PPT algorithms* $(\mathsf{Gen}_{\mathsf{Obf}}, \mathsf{Obf}, \mathsf{Eval})$ *such that the following holds:*

- **Correctness:** *For all* $\{\Pi_\lambda\}_{\lambda\in\mathbb{N}} \in \mathcal{C}_{\log(n),s,d}$*, there exists a negligible function* μ *such that for all* $\lambda \in \mathbb{N}$*, all* $\mathbf{x} \in \{0,1\}^{n(\lambda)}$*:*

$$\Pr[\mathsf{pp} \leftarrow \mathsf{Gen}_{\mathsf{Obf}}(1^\lambda), \widetilde{\Pi} \leftarrow \mathsf{Obf}(\mathsf{pp}, \Pi_\lambda) : \mathsf{Eval}(\mathsf{pp}, \widetilde{\Pi}, \mathbf{x}) = \Pi(\mathbf{x})] \geq 1 - \mu(\lambda)$$

- **Succinctness:** *For all* $\{\Pi_\lambda\}_{\lambda\in\mathbb{N}} \in \mathcal{C}_{\log(n),s,d}$*, all* $\lambda \in \mathbb{N}$*, all* pp *in the support of* $\mathsf{Gen}_{\mathsf{Obf}}(1^\lambda)$*, all* $\widetilde{\Pi}$ *in the support of* $\mathsf{Obf}(\mathsf{pp}, \Pi_\lambda)$*, we have that* $\left|\widetilde{\Pi}\right| \leq n(\lambda)^{1-\varepsilon} \cdot p(\lambda, s(\lambda), d(\lambda))$

- **IND-security:** *For all sequences* $\{\Pi_0^\lambda\}_{\lambda\in\mathbb{N}}, \{\Pi_1^\lambda\}_{\lambda\in\mathbb{N}} \in \mathcal{C}_{\log(n),s,d}$ *such that for all* $\lambda \in \mathbb{N}$*,* Π_0^λ *and* Π_1^λ *are functionally equivalent circuit, the following ensembles are computationally indistinguishable:*

$$\left\{\mathsf{pp} \leftarrow \mathsf{Gen}_{\mathsf{Obf}}(1^\lambda), \widetilde{\Pi} \leftarrow \mathsf{Obf}(\mathsf{pp}, \Pi_\lambda^0) : (\mathsf{pp}, \widetilde{\Pi})\right\}_{\lambda\in\mathbb{N}}$$
$$\left\{\mathsf{pp} \leftarrow \mathsf{Gen}_{\mathsf{Obf}}(1^\lambda), \widetilde{\Pi} \leftarrow \mathsf{Obf}(\mathsf{pp}, \Pi_\lambda^1) : (\mathsf{pp}, \widetilde{\Pi})\right\}_{\lambda\in\mathbb{N}}$$

The following theorem from [LPST16] connects $Xi\mathcal{O}$ (with pre-processing) with $i\mathcal{O}$ assuming the LWE assumption (we formally define the LWE assumption in Definition 3.4).

Theorem 2.4. *Assume the existence of a subexponentially-secure* $Xi\mathcal{O}$ *for* $\mathsf{P}^{\log}/\mathsf{poly}$*, and assume subexponential security of the LWE assumption. Then there exists an (subexponentially-secure)* $i\mathcal{O}$ *for* P/poly*.*

2.5 Definition of Public-Key Encryption

We start by recalling the definition of public key encryption (PKE). For our purposes, we will consider PKE in a Common Reference String (CRS) model, where we first generate a CRS, and next, the key generation algorithm will take the CRS as input. This added generality will be useful to capture scenarios where multiple encryption schemes will be operating over the same ring $\mathbb{Z}_N$—this ring can be specified in the CRS.

Definition 2.5 (Public-Key Encryption)**.** *A Public-Key Encryption (PKE) scheme is a tuple of PPT algorithms* $(\mathsf{CRSgen}, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *where:*

- $\mathsf{CRSgen}(1^\lambda)$*: given as input the security parameter* $\lambda \in \mathbb{N}$*, it outputs a common reference string* crs*.*
- $\mathsf{Gen}(\mathsf{crs})$*: given as input* crs*, it outputs the pair* $(\mathsf{pk}, \mathsf{sk})$*.*
- $\mathsf{Enc}_{\mathsf{pk}}(m; r)$*: given as input the public key* pk*, a message* $m \in \{0,1\}^*$ *and some randomness* $r \leftarrow_R \{0,1\}^\infty$[11]*, it outputs a ciphertext* ct*.*
- $\mathsf{Dec}_{\mathsf{sk}}(\mathsf{ct})$*: given as input the secret key* sk *and a ciphertext* ct*, it deterministically outputs a plaintext.*

We furthermore require these algorithms to satisfy the following correctness *condition: for all* $\lambda \in \mathbb{N}$*, all* crs *in the support of* $\mathsf{CRSgen}(1^\lambda)$*, all pairs* $(\mathsf{pk}, \mathsf{sk})$ *in the support* $\mathsf{Gen}(\mathsf{crs})$*, all messages* $m \in \{0,1\}^*$*, all ciphertexts* ct *in the support of* $\mathsf{Enc}_{\mathsf{pk}}(m)$*, we have:*

$$\mathsf{Dec}_{\mathsf{sk}}(\mathsf{ct}) = m.$$

2.6 Definition of Linearly-Homomorphic Encryption

Definition 2.6 (Linearly-Homomorphic Encryption)**.** *For any polynomial* $\ell(\cdot)$*, a PKE scheme* $(\mathsf{CRSgen}, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is said to be a Linearly-Homomorphic Encryption (LHE) with plaintext size* $\ell(\cdot)$*, if there exists a PPT algorithm* Add *such that the following holds:*

- *For all* $\lambda \in \mathbb{N}$*, all* crs *in the support of* $\mathsf{CRSgen}(1^\lambda)$*, all* $(\mathsf{pk}, \mathsf{sk})$ *in the support of* $\mathsf{Gen}(\mathsf{crs})$*, the public key* pk *contains a message space* $(\mathbb{A}_{\mathsf{pk}}, +)$*, which is an Abelian group of size* $|\mathbb{A}| > 2^{\ell(\lambda)}$*.*
- *For all* $\lambda \in \mathbb{N}$*, all* crs *in the support of* $\mathsf{CRSgen}(1^\lambda)$*, all* $(\mathsf{pk}, \mathsf{sk})$ *in the support of* $\mathsf{Gen}(\mathsf{crs})$*, all messages* $m_1, m_2 \in \mathbb{A}_{\mathsf{pk}}$*, all ciphertexts* $\mathsf{ct}_1, \mathsf{ct}_2$ *in the support of* $\mathsf{Enc}_{\mathsf{pk}}(m_1), \mathsf{Enc}_{\mathsf{pk}}(m_2)$ *respectively, the algorithm* $\mathsf{Add}(\mathsf{pk}, \mathsf{ct}_1, \mathsf{ct}_2)$ *determinsitically outputs a ciphertext in the support of* $\mathsf{Enc}_{\mathsf{pk}}(m_1 + m_2)$*, where the addition is performed in* $\mathbb{A}_{\mathsf{pk}}$*.*

2.7 Definition of Fully Homomorphic Encryption

Definition 2.7 (Fully-Homomorphic Encryption)**.** *A PKE scheme* $(\mathsf{CRSgen}, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is said to be a Fully-Homomorphic Encryption (FHE) scheme for depth* $\delta(\cdot)$ *circuits if there exists a PPT algorithm* Eval *such that for all* $\lambda \in \mathbb{N}$*, all* crs *in the support of* $\mathsf{CRSgen}(1^\lambda)$*, all pairs* $(\mathsf{pk}, \mathsf{sk})$ *in the support of* Gen*, all* $n \in \mathbb{N}$*, all messages* $m_1, \ldots, m_n \in \{0,1\}$*, all ciphertexts* $\mathsf{ct}_1, \ldots, \mathsf{ct}_n$ *in the support of* $\mathsf{Enc}_{\mathsf{pk}}(m_1), \ldots, \mathsf{Enc}_{\mathsf{pk}}(m_n)$ *respectively, all circuits* $f : \{0,1\}^n \to \{0,1\}$ *of depth at most* $\delta(\lambda)$*,* $\mathsf{Eval}(\mathsf{pk}, f, \mathsf{ct}_1, \ldots, \mathsf{ct}_n)$ *deterministically outputs an evaluated ciphertext* ct_f *such that* $\mathsf{Dec}_{\mathsf{sk}}(\mathsf{ct}_f) = f(m_1, \ldots, m_n)$*.*

Note that the depth of the circuit that are homomorphically evaluated is a priori bounded by $\delta(\lambda)$ for a polynomial δ (that is, we consider the case of *leveled* FHE). The arity of the evaluated circuits (denoted by n above), however, is unbounded. The FHE we will be using — namely, from [GSW13] — natively supports arithmetic circuits (with addition and multiplication gates), which capture Boolean circuits.

[11] As usual, since all algorithms are PPT we really only need to consider a finite prefix of $\{0,1\}^\infty$ to define the uniform distribution.

2.8 Leakage-resilient and Circular Security

We recall the standard definition of CPA-security for encryption schemes; we furthermore generalize it to a notion of $\mathcal{O}$-leakage resilient security, which extends the standard definition by also providing the attacker with access to a leakage oracle $\mathcal{O}$ receiving the message $\mathbf{m}^\star$ being encrypted, and the randomness $\mathbf{r}$ under which it is encrypted. Our notion of $\mathcal{O}$ leakage-resilience restricts to attackers that only make "valid" leakage queries, where a query is said to be valid if the oracle does not return $\perp$ in response to it. In more detail, to "win" in the security game, the attacker $\mathcal{A}$ must (a) correctly guess which among two message $\mathbf{m}^0, \mathbf{m}^1$ is being encrypted, while (b) not having made any queries to $\mathcal{O}$ on which $\mathcal{O}$ returns $\perp$.

Definition 2.8 ($\mathcal{O}$-leakage resilient security)**.** *We say that a public-key encryption scheme* $\mathcal{PKE} = (\mathsf{CRSgen}, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is $\mathcal{O}$-leakage resilient secure if for all stateful nuPPT adversaries $\mathcal{A}$, there exists some negligible function $\mu(\cdot)$ such that for all $\lambda \in \mathbb{N}$,* $\Pr[\mathsf{Exp}_{\lambda,\mathcal{A}} = 1] \leq 1/2 + \mu(\lambda)$*, where the experiment* $\mathsf{Exp}^{\mathcal{PKE}}_{\lambda,\mathcal{A}}$ *is defined as follows:*

$$\mathsf{Exp}^{\mathcal{PKE}}_{\lambda,\mathcal{A}} = \left\{ \begin{array}{l} \mathsf{crs} \leftarrow \mathsf{CRSgen}(1^\lambda), (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(\mathsf{crs}) \\ (\mathbf{m}^0, \mathbf{m}^1) \leftarrow \mathcal{A}(\mathsf{pk}), b \leftarrow \{0,1\} \\ \mathbf{m}^\star = \mathbf{m}^b, \mathbf{r} \leftarrow_{\mathsf{R}} \{0,1\}^\infty \\ \mathsf{ct} = \mathsf{Enc}_{\mathsf{pk}}(\mathbf{m}^\star; \mathbf{r}), b' \leftarrow \mathcal{A}^{\mathcal{O}(\mathbf{m}^\star, \mathbf{r})}(\mathsf{ct}) \\ \textit{Return 1 if } |\mathbf{m}^0| = |\mathbf{m}^1|,\ b' = b \textit{ and } \mathcal{O} \textit{ did not return } \perp\textit{; 0 otherwise.} \end{array} \right\}$$

We say that $\mathcal{PKE}$ is simply secure *if the above holds when we do not give $\mathcal{A}$ access to an oracle.*

We will also consider a 1-circular secure variant of $\mathcal{O}$-leakage resilient security, which is identically defined except the message $\mathbf{m}^\star$ is defined as $\mathsf{sk}||\mathbf{m}^b$. In other words, we require indistinguishability of $\mathbf{m}^0$ and $\mathbf{m}^1$ in the presence of an encryption of the secret key sk.

Definition 2.9 ($\mathcal{O}$-leakage resilient circular security)**.** *We say that* $\mathcal{O}$-leakage resilient 1-circular security *holds for $\mathcal{PKE}$ if Definition 2.8 holds w.r.t. $\mathcal{PKE}$ with the exception that $\mathbf{m}^\star$ is defined as $\mathsf{sk}||\mathbf{m}^b$ (instead of $\mathbf{m}^b$).*

We finally state the 1CIRC assumption that we will rely on in our main theorem.

Definition 2.10 (1CIRC assumption)**.** *We say that the* (subexponential) $\mathsf{1CIRC}^{\mathcal{O}}$ assumption holds w.r.t $\mathcal{PKE}$ *if the following holds: if $\mathcal{PKE}$ is (subexponentially) $\mathcal{O}$-leakage resilient secure, then (subexponential) $\mathcal{O}$-leakage resilient 1-circular security holds for $\mathcal{PKE}$.*

A Note of the Definition of Circular Security. Let us remark that while the above definition of circular security (i.e. indistinguishability of encrypted messages in the presence of an encryption of the secret key) is the most direct way of capturing the needs for circular security in applications (think e.g., of encrypted disk space), an alternative definition is also commonly used in the literature: it requires (a) standard security, and (b) indistinguishability of encryptions of sk and 0^λ. We want to emphasize that whereas in the standard setting—*without a leakage oracle*—this alternative definition implies the circular security notion we gave, this is no longer seems to be true in the oracle-enhanced setting. To see this, consider an oracle that given $\mathbf{m}^\star, \mathbf{r}$, outputs $\perp$ to any query in case $\mathbf{m}^\star$ does not contain a valid secret key, and consider some encryption scheme (e.g., GSW) for which it seems hard to come up with encryptions of the secret key (given just the public key). Leakage queries can never be useful to distinguish encryptions of sk and 0^λ—as the oracle will output $\perp$ with probability negligibly close to $1/2$ and thus the attacker's advantage is negligible—so $\mathcal{O}$-leakage resilient circular security is equivalent to plain (i.e. without an oracle) circular security. Yet, leakage queries w.r.t.

such an oracle may be useful when considering indistinguishability between $\mathsf{sk}||\mathbf{m}_0$ and $\mathsf{sk}||\mathbf{m}_1$. In fact, for the particular leakage queries that we will be relying on, this phenomena does occur: they are valid in case $\mathbf{m}^\star$ is of the form $\mathsf{sk}||\mathbf{m}$ where sk is a valid secret key, but they are invalid if sk is not a valid secret key. For this reason, we directly formalize circular security as indistinguishability of encrypted messages in the presence of an encrypted secret key.

3 Shielded Randomness Leakage Security of GSW

In this section, we define our notion of Shielded Randomness Leakage (SRL) security, which corresponds to $\mathcal{O}$-leakage resilience security for a particular leakage oracle $\mathcal{O}$. Then, we prove the GSW FHE is SRL secure under the LWE assumption.

3.1 Definition of Shielded Randomness Leakage Security

To define our notion of SRL security, we focus on FHE schemes that satisfy the following properties.

3.1.1 Batch correctness

This property states that decryption of evaluated ciphertexts solely consists of computing the inner product of the evaluated ciphertext with the secret key (both of which are vectors), then rounding. Also, a single scalar obtained by decryption can encode many output bits of the evaluated function. That is, we consider FHE scheme where the crs contains a modulus N_{crs} such that decryption of an evaluated ciphertext yields a scalar in $\mathbb{Z}_{N_{\mathsf{crs}}}$. Our definition of FHE is flexible w.r.t. the choice of the modulus N_{crs}, which we can afford since the LWE assumption holds for essentially any (large enough) modulus. As observed in [Mic19, BDGM19, BDGM20a], most existing FHE schemes can fit this framework.

Definition 3.1 (Batch correctness)**.** *For all poynomials δ, an FHE scheme* $(\mathsf{CRSgen}, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ *for depth-δ circuits satisfies batch correctness if there exist a PPT* Eval' *and a polynomial σ such that following holds:*

- *For all $\lambda \in \mathbb{N}$, all* crs *in the support of* $\mathsf{CRSgen}(1^\lambda)$ *contain a modulus $N_{\mathsf{crs}} \in \mathbb{N}$; for all* $(\mathsf{pk}, \mathsf{sk})$ *in the support of* $\mathsf{Gen}(\mathsf{crs})$*, we have:* pk *contains $B_{\mathsf{pk}} \in \mathbb{N}$ such that $N_{\mathsf{crs}} \geq 2^\lambda B_{\mathsf{pk}}$; the secret key is of the form:* $\mathsf{sk} \in \mathbb{Z}^{\sigma(\lambda)}$.
- *For all $\lambda \in \mathbb{N}$, all* crs *in the support of* $\mathsf{CRSgen}(1^\lambda)$*, all* $(\mathsf{pk}, \mathsf{sk})$ *in the support of* $\mathsf{Gen}(\mathsf{crs})$*, all arities $\nu \in \mathbb{N}$, all messages $m_1, \ldots, m_\nu \in \{0,1\}$, all depth-$\delta(\lambda)$ circuits f of arity ν, all ciphertexts* ct_i *in the support of* $\mathsf{Enc}_{\mathsf{pk}}(m_i)$ *for all $i \in [\nu]$, all scaling factors $\omega < \log(N_{\mathsf{crs}})$, the algorithm* $\mathsf{Eval}'(\mathsf{pk}, f, \omega, \mathsf{ct}_1, \ldots, \mathsf{ct}_\nu)$ *deterministically outputs an evaluated ciphertext* $\mathsf{ct}_f \in \mathbb{Z}_{N_{\mathsf{crs}}}^{\sigma(\lambda)}$ *such that:*
$$\mathsf{sk}^\top \mathsf{ct}_f = 2^\omega f(\mathbf{m}) + \mathsf{noise}_f \in \mathbb{Z}_{N_{\mathsf{crs}}},$$
with $|\mathsf{noise}_f| < B_{\mathsf{pk}}$.

Note that one can recover the value $f(\mathbf{m}) \in \mathbb{Z}_{N_{\mathsf{crs}}}$ when using the scaling factor $\omega = \lceil \log(B_{\mathsf{pk}}) \rceil + 1$. That is, we can define $\mathsf{Eval}(\mathsf{pk}, f, \mathsf{ct}_1, \ldots, \mathsf{ct}_\nu) = \mathsf{Eval}'(\mathsf{pk}, f, \lceil \log(B_{\mathsf{pk}}) \rceil + 1, \mathsf{ct}_1, \ldots, \mathsf{ct}_\nu)$.

3.1.2 Randomness homomorphism

This property states that it is possible to homomorphically evaluates a circuit f not only on the ciphertexts, but also the randomness used by the ciphertexts. The resulting evaluated randomness $\mathbf{r}_f$ belongs to a noisy randomness space $\mathcal{R}^\star$ — typically the fresh randomness comprises noises, and the evaluated randomness consists of larger-magnitude noises. The encryption algorithm $\mathsf{Enc}^\star$ is essentially the same as Enc except it operates on the evaluated (noisier) randomness. The ciphertext obtained by first evaluating the randomness, then using the noisy encryption algorithm $\mathsf{Enc}^\star$ is the same as obtained by directly evaluating the original ciphertexts.

Definition 3.2 (Randomness homomorphism)**.** *An FHE scheme $\mathcal{FHE} = (\mathsf{CRSgen}, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ for depth-δ circuits that satisfies batch correctness (defined above) also satisfies randomness homomorphism if there exists a sequence of noisy randomness spaces $\{\mathcal{R}^\star_\lambda\}_{\lambda\in\mathbb{N}}$, and the following additional PPT algorithms:*

- $\mathsf{Eval}_{\mathsf{rand}}(\mathsf{pk}, f, \mathbf{r}, \mathbf{m})$*: given as input the public key* pk*, a depth-$\delta(\lambda)$ circuit f of arity ν, random coins $\mathbf{r} = (\mathbf{r}_1, \ldots, \mathbf{r}_\nu)$ where for all $i \in [\nu]$, $\mathbf{r}_i \in \{0,1\}^\infty$, and messages $\mathbf{m} \in \{0,1\}^\nu$, it deterministically outputs an evaluated randomness $\mathbf{r}_f \in \mathcal{R}^\star_\lambda$.*
- $\mathsf{Enc}^\star_{\mathsf{pk}}(m; \mathbf{r}^\star)$*: given as input the public key* pk*, a message $m \in \mathbb{Z}_{N_{\mathsf{crs}}}$ and the randomness $\mathbf{r}^\star \in \mathcal{R}^\star$, it outputs a noisy ciphertext* $\mathsf{ct}^\star$*.*

We furthermore require these algorithms to satisfy the following condition: for every $\lambda \in \mathbb{N}$, all crs *in the support of* $\mathsf{CRSgen}(1^\lambda)$*, all pairs* $(\mathsf{pk}, \mathsf{sk})$ *in the support of* $\mathsf{Gen}(\mathsf{crs})$*, all $\nu \in \mathbb{N}$, all depth-$\delta(\lambda)$ circuits f of arity ν, all messages $m_i \in \{0,1\}$, all randomness $\mathbf{r}_i \in \{0,1\}^\infty$ for $i \in [\nu]$, denoting* $\mathsf{ct}_i = \mathsf{Enc}_{\mathsf{pk}}(m_i; \mathbf{r}_i)$ *and* $\mathbf{r}_f = \mathsf{Eval}_{\mathsf{rand}}(\mathsf{pk}, f, \mathbf{r}, \mathbf{m})$*, we have $\mathbf{r}_f \in \mathcal{R}^\star_\lambda$ and:*

$$\mathsf{Eval}'(\mathsf{pk}, f, 0, \mathsf{ct}_1, \ldots, \mathsf{ct}_\nu) = \mathsf{Enc}^\star_{\mathsf{pk}}(f(\mathbf{m}); \mathbf{r}_f).$$

3.1.3 Shielded Randomness-Leakage security

We proceed to formally define shielded randomness leakage (SRL) security for randomness homomorphic FHEs with batch correctness. SRL security will be defined as $\mathcal{O}$-leakage resilient security for a particular leakage oracle $\mathcal{O}_{\mathsf{SRL}}$ that given a message $\mathbf{m}^\star$ and randomness $\mathbf{r}$, allows the attacker $\mathcal{A}$ to ask to see a "shielded" version of the homomorphically evaluated randomness $\mathbf{r}_f$ for any function f *for which $\mathcal{A}$ knows the output* $f(\mathbf{m}^\star)$. To make sure the attacker can only query the oracle with functions on which it knows the output, we require the atacker to also provide the output α, and the oracle outputs $\perp$ if $f(\mathbf{m}^\star) \neq \alpha$ (and thus, by the definition of $\mathcal{O}$-leakage resilient security, the attacker fails if it ever picks a function for which it does not know the output).

To formalize the SRL oracle, we restrict ourselves to FHE where the noisy randomness consists of integer vectors. That is, there exists a polynomial $t(\cdot)$ such that the sequence $\{\mathcal{R}^\star_\lambda\}_{\lambda\in\mathbb{N}}$ is such that for all $\lambda \in \mathbb{N}$, $\mathcal{R}^\star_\lambda \subseteq \mathbb{Z}^{t(\lambda)}$. Henceforth, we denote by $\mathbf{r}_1 + \mathbf{r}_2 \in \mathcal{R}^\star_\lambda$ and $\mathbf{r}_1 - \mathbf{r}_2 \in \mathcal{R}^\star_\lambda$ the addition and subtraction in $\mathbb{Z}^{t(\lambda)}$. We denote $\mathcal{R}^\star_\lambda$ by $\mathcal{R}^\star$ for simplicity.

Definition 3.3 (SRL security)**.** *An FHE scheme $\mathcal{FHE}$ for depth δ circuits satisfying randomness homomorphism is said to be (circular) SRL-secure if it is $\mathcal{O}^{\mathcal{FHE}}_{\mathsf{SRL}}$-leakage resilient (1-circular) secure for the following oracle $\mathcal{O}^{\mathcal{FHE}}_{\mathsf{SRL}}$, where* $\mathsf{Eval}_{\mathsf{rand}}$ *and* $\mathsf{Enc}^\star$ *are the algorithms guaranteed to exist by the definition of randomness homomorphism.*

$\underline{\mathcal{O}_{\mathsf{SRL}}^{\mathcal{FHE}}(\mathbf{m}^\star, \mathbf{r})}$:
$\mathbf{r}^\star \leftarrow_{\mathsf{R}} \mathcal{R}^\star$, $\mathsf{ct}^\star = \mathsf{Enc}_{\mathsf{pk}}^\star(0; \mathbf{r}^\star)$
$(f, \alpha) \leftarrow \mathcal{A}(\mathsf{ct}^\star)$
$\mathbf{r}_f = \mathsf{Eval}_{\mathsf{rand}}(\mathsf{pk}, f, \mathbf{r}, \mathbf{m}^\star)$.
If $f(\mathbf{m}^\star) = \alpha$ *and* f *is of depth at most* δ, *then* $\mathsf{leak} = \mathbf{r}^\star - \mathbf{r}_f \in \mathcal{R}^\star$.
Otherwise, $\mathsf{leak} = \perp$. *Return* leak.

Roughly speaking, given a message $\mathbf{m}^\star$ and randomness $\mathbf{r}$, the oracle $\mathcal{O}_{\mathsf{SRL}}^{\mathcal{FHE}}$ samples fresh random coins $\mathbf{r}^\star$ from which it generates a noisy encryption of zero, that is sent to the adversary. The adversary next chooses a circuit f and a value $\alpha \in \mathbb{Z}$. The oracle then checks that $f(\mathbf{m}^\star) = \alpha$, upon which it returns the evaluated randomness "shielded" with the randomness $\mathbf{r}^\star$; otherwise, it outputs $\perp$ and in this case, the attacker fails.

In the concrete FHE we consider from [GSW13], the randomness leakage corresponds to the randomness obtained from homomorphically subtracting the evaluated challenge ciphertext from $\mathsf{Enc}_{\mathsf{pk}}^\star(0; \mathbf{r}^\star)$. Revealing such leakage allows the adversary to decrypt and recover the value $0 - f(\mathbf{m}^\star)$. This is why we only allow the attacker to request leakage functions f for which it knows the output $f(\mathbf{m}^\star)$. Whenever the scheme $\mathcal{FHE}$ is clear from context, we simply write $\mathcal{O}_{\mathsf{SRL}}$ to denote $\mathcal{O}_{\mathsf{SRL}}^{\mathcal{FHE}}$.

A note on the falsifiability and interactivity of (circular) SRL security. We note that both SRL and circular SRL security of an FHE is a simple and natural *interactive* falsifiable assumption about the FHE: the assumption is defined as an interactive security game involving a PPT challenger $\mathcal{C}$, with a threshold of $1/2$ (i.e. the attacker needs to win with probability non-negligibly higher than $1/2$). Let us make some additional observations about this assumption:

- Let us first point out that for our actual application, we only need to rely on a relaxed form of SRL security where $\mathcal{A}$ sends all its queries in parallel. Thus, such a "parallel" SRL assumption can be captured as a 6-round security game (1) $\mathcal{A}$ first gets the public key, (2) $\mathcal{A}$ picks the messages $\mathbf{m}^0, \mathbf{m}^1$, (3) $\mathcal{A}$ gets the encryptions of $\mathbf{m}^b$ and the shields, (4) $\mathcal{A}$ selects the parallel leakage queries $\{f_i, \alpha_i\}$, (5) $\mathcal{A}$ gets back the (shielded) randomness $\{\mathbf{r}_{f_i}\}$ and (6) $\mathcal{A}$ finally makes a guess for b. In fact, we actually do not need CPA security for adaptively chosen messages so we can compress it to a 5-round assumption. We write down this concrete 5-round assumption that suffices for us in Appendix **??**.

- Next, let us note that for our application, we only need SRL security for: (1) *short* messages $\mathbf{m}^0, \mathbf{m}^1$, of length λ^ε, for some $\varepsilon \in (0, 1)$, (2) a query-selecting mechanism that has some fixed polynomial running time, and whose description size is only $\lambda^\varepsilon + O(1)$. For such applications, we can stipulate a *non-interactive* SRL-security game: the challenger picks *random* messages $\mathbf{m}^0, \mathbf{m}^1$, and a *random* query-selecting machine TM M, both of size $O(\lambda^\varepsilon)$. Next, it checks that M generates valid queries (within the a-priori fixed time-bound) w.r.t. to *both* $\mathbf{m}^0, \mathbf{m}^1$ (or in the case of circular security w.r.t. $\mathsf{sk}||\mathbf{m}^0$ and $\mathsf{sk}||\mathbf{m}^1$). If so, the challenger gives the attacker the public key, an encryption of $\mathbf{m}^\star$, the shields and the SRL leakage. The attacker wins if it correctly guesses b. If not (i.e. some of the SRL queries were invalid), the same information *excluding the SRL leakage* is sent to the attacker.

  If this non-interactive SRL-assumption is $2^{O\left(\lambda^{\varepsilon'}\right)}$-hard, where $\varepsilon' > \varepsilon$, it implies that it is still $2^{O\left(\lambda^{\varepsilon'}\right)}$-hard for *every* (short) choices of $\mathbf{m}^0, \mathbf{m}^1$ and M by a union bound over $\mathbf{m}^0, \mathbf{m}^1, M$, and this is exactly what is needed for our $Xi\mathcal{O}$ proof, as the query-selecting machine selects leakage queries that are valid w.r.t. $\mathsf{sk}||\mathbf{m}^0$ and $\mathsf{sk}|\mathbf{m}^1$ (with high probability).

While the above discussion shows that we could have presented a relatively simple *non-interactive* (and falsifiable) circular SRL-assumption (which we can prove holds w.r.t. GSW based on LWE in the non-circular setting), in our opinion, doing so does not elucidate the assumption on a qualitative level. Rather, we have chosen to present the circular SRL-assumption in a more general and stronger form as we believe this interactive version is: (1) *more natural*—it captures a very natural (in our eyes) security game for an FHE, and (2) it is *still secure in the non-circular setting* (based on LWE), and (3) due to the fact that the assumption is *stronger and simpler*, it becomes easier to attack (and conversely, the absence of attacks would inspire more confidence).

Let us furthermore note that for our proof, we only need to consider very specific SRL leakage queries and thus an alternative way of making the assumption "technically weaker" is to restrict to those types of queries, but we believe that would just make the assumption more ad-hoc, without adding any real reason for increased confidence in the security.

3.2 SRL Security of the GSW FHE from LWE

We now recall the FHE scheme from [GSW13], whose security relies on the LWE assumption. The variant we present uses a large modulus to permit batching many output bits in a single scalar. We prove the GSW scheme is SRL-secure (as per Definition 3.3) under the LWE assumption.

3.2.1 Learning With Error Assumption

We recall the Learning with Error (LWE) assumption with subexponential modulus-to-noise ratio. In [Reg05], Regev showed that solving the LWE problem with modulus q, dimension κ, arbitrary number of samples m, and discrete Gaussian distribution χ of standard deviation $\sigma = \alpha q \geq 2\sqrt{\kappa}$ (this is the distribution over $\mathbb{Z}$ that follows the normal distribution of standard deviation σ, and it is such that $\Pr[e \leftarrow \chi : |e| > \sigma\sqrt{\log(\kappa)}] \in 2^{-\Omega(\kappa)}$) is at least as hard as quantumly approximating the shortest independent vector problem (SIVP) to within an approximation factor $\gamma = \widetilde{\mathcal{O}}(\kappa/\alpha)$ in the *worst case* κ-dimensional lattices. His result only applied to every modulus q that is a prime power, or a product of small (poly-size) distinct primes. Later, in [PRS17], the result was generalized to any modulus q.

As typical, we choose a noise-to-modulus ratio $\alpha = 2^{-\kappa^c}$ for a constant $c \in (0, 1)$, which corresponds to the SIVP problem with an approximation factor $\gamma = \widetilde{\mathcal{O}}(\kappa \cdot 2^{\kappa^c})$, which is believed to be intractable for c small enough.

Originally, the LWE assumption was defined for uniformly random secrets; however, several works [Mic01, ACPS09] showed that LWE is no easier if the secret is drawn from the noise distribution, which is the variant we will use. Finally, we use an noise distribution that is bounded with probability 1 (as opposed to all but negligible probability, as it is the case for the discrete Gaussian distribution that is commonly used). This mild strengthening is not fundamental but will make our definitions easier to work with (e.g. correctness will hold with probability 1). That is, we rely on the following LWE assumption.

Definition 3.4 (LWE assumption [Reg05, PRS17])**.** *For all sequences $q \in 2^{\mathsf{poly}(\kappa)}$, all ensembles χ of efficiently sampleable distributions over $\mathbb{Z}$, we say that (subexponential) security of the LWE assumption holds w.r.t. the sequence q and the ensemble χ if for all polynomials $m(\cdot)$, the following ensembles are (subexponentially) computationally indistinguishable:*

$$\left\{\mathbf{A} \leftarrow_{\mathsf{R}} \mathbb{Z}_{q_\kappa}^{m(\kappa)\times\kappa}, \mathbf{s} \leftarrow \chi_\kappa^\kappa, \mathbf{e} \leftarrow \chi_\kappa^{m(\kappa)}, \mathbf{z} = \mathbf{A}\mathbf{s} + \mathbf{e} \in \mathbb{Z}_{q_\kappa}^{m(\kappa)} : (\mathbf{A}, \mathbf{z})\right\}_{\kappa\in\mathbb{N}}.$$
$$\left\{\mathbf{A} \leftarrow_{\mathsf{R}} \mathbb{Z}_{q_\kappa}^{m(\kappa)\times\kappa}, \mathbf{z} \leftarrow_{\mathsf{R}} \mathbb{Z}_{q_\kappa}^{m(\kappa)} : (\mathbf{A}, \mathbf{z})\right\}_{\kappa\in\mathbb{N}}.$$

We say the (subexponential) security of the LWE assumption holds if there exists a constant $c \in (0,1)$ such that for all sequences $q \in 2^{\mathsf{poly}(\kappa)}$ and all polynomials B such that for all $\kappa \in \mathbb{N}$, the following holds:

- $B(\kappa) \geq 2\sqrt{\kappa \log(\kappa)}$
- $B(\kappa) \geq q_\kappa 2^{-\kappa^c}$

there exists a B-bounded ensemble χ of efficiently sampleable distributions over $\mathbb{Z}$, such that (subexponential) security of the LWE assumption holds w.r.t. q, χ.

3.2.2 The GSW scheme

We recall the FHE from [GSW13]. We present the leveled variant (without bootstrapping), which is parameterized by a polynomial δ that bounds the depth of the circuits that can be homomorphically evaluated. Its security relies on the LWE assumption with a subexponential modulus-to-noise ratio. We denote the scheme by GSW_δ.

The key generation algorithm we describe works when given as input any CRS that contains a modulus N, that will be used by the scheme. Apart from being sufficiently large to ensure correctness (that is, larger than the noise magnitude obtained when evaluating circuits of depth at most δ), no property is required from the modulus.

This way, the key generation algorithm can be fed with the CRS generated as in the GSW FHE scheme, and recalled here, but also with a CRS that describes the public key of a Linearly Homomorphic Encryption scheme that performs linearly operation over $\mathbb{Z}_N$. This ensures compatibility: both schemes can operate on the same ring $\mathbb{Z}_N$.

Notations. For all polynomials δ, we denote by b_δ the polynomial such that for all poynomially-bounded ensemble χ, all polynomials κ, all $\lambda \in \mathbb{N}$, the noise obtained from homomorphically evaluating circuits of depth at most $\delta(\lambda)$ on GSW ciphertexts generated with LWE noise sampled from the distribution χ_λ and LWE secret of dimension $\kappa(\lambda)$, is upper bounded by $2^{b_\delta(\lambda)}$.

- $\underline{\mathsf{CRSgen}}(1^\lambda)$:
  Let $N = 2^{2\lambda + b_\delta(\lambda)}$. The algorithm outputs $\mathsf{crs} = N$.

- $\underline{\mathsf{Gen}}(\mathsf{crs})$:
Given as input crs which contains a modulus $N \geq 2^{2\lambda + b_\delta(\lambda)}$, it chooses a sequence $\{q_n\}_{n\in\mathbb{N}}$, a B_χ-bounded ensemble $\{\chi_n\}_{n\in\mathbb{N}}$ for a polynomial B_χ and a polynomial κ such that LWE holds w.r.t. q and χ, and $q_{\kappa(\lambda)} = N$ (by the LWE assumption, given in Definition 3.4, we know such parameters exist). We abuse notations and write $\kappa = \kappa(\lambda)$, $\chi = \chi_{\kappa(\lambda)}$ and $B_\chi = B_\chi(\kappa(\lambda))$ from here on. The algorithm sets $w = (\kappa+1)\lceil \log(N) \rceil$, $m = 2(\kappa+1)\lceil \log(N) \rceil + 2\lambda$, $B^\star = 2^\lambda (w+1)^\delta \lceil \log(N) \rceil$ and $B = B_\chi (w+1)^\delta \lceil \log(N) \rceil m$. Note that we have $N \geq 2^{2\lambda} B$.

It samples $\mathbf{A} \leftarrow_{\mathsf{R}} \mathbb{Z}_N^{\kappa \times m}$, $\mathbf{s} \leftarrow \chi^\kappa$, $\mathbf{e} \leftarrow \chi^m$, $\mathbf{g} = (1, 2, \ldots, 2^{\lceil \log(N) \rceil - 1}) \in \mathbb{Z}_N^{\lceil \log(N) \rceil}$, $\mathbf{G} = \mathbf{g}^\top \otimes \mathrm{Id} = \begin{pmatrix} \mathbf{g}^\top & 0 & \cdots \\ 0 & \mathbf{g}^\top & \\ \vdots & & \ddots \end{pmatrix} \in \mathbb{Z}_N^{(\kappa+1)\times w}$ where $\mathrm{Id} \in \mathbb{Z}_N^{(\kappa+1)\times(\kappa+1)}$ denotes the identity matrix, $\mathbf{U} = \begin{pmatrix} \mathbf{A} \\ \mathbf{s}^\top \mathbf{A} + \mathbf{e}^\top \end{pmatrix} \in \mathbb{Z}_N^{(\kappa+1)\times m}$. It sets $\mathsf{pk} = (B, \mathbf{U}, \mathbf{G})$, and $\mathsf{sk} = (-\mathbf{s}, 1) \otimes \mathbf{g} \in \mathbb{Z}_N^w$. The parameters define the noisy randomness space $\mathcal{R}^\star = [-B^\star, B^\star]^m$. It outputs $(\mathsf{pk}, \mathsf{sk})$.

• $\underline{\mathsf{Enc}(\mathsf{pk}, m)}$:
Given the public pk, a message $m \in \{0,1\}$, it samples the randomness $\mathbf{R} \leftarrow_{\mathsf{R}} [-1,1]^{m \times w}$ and outputs the ciphertext $\mathsf{ct} = \mathbf{UR} + m\mathbf{G} \in \mathbb{Z}_N^{(\kappa+1)\times w}$. For any $\mathbf{m} \in \{0,1\}^n$, we denote by $\mathsf{Enc}_{\mathsf{pk}}(\mathbf{m}; \mathbf{r})$ the concatenation of the encryptions $\mathsf{Enc}_{\mathsf{pk}}(m_1; \mathbf{R}_1), \ldots, \mathsf{Enc}_{\mathsf{pk}}(m_n; \mathbf{R}_n)$.

• $\underline{\mathsf{Eval}(\mathsf{pk}, f, \mathsf{ct}_1, \ldots, \mathsf{ct}_\nu)}$:
Given the public key pk, a depth-$\delta(\lambda)$ arithmetic $f : \{0,1\}^\nu \to \{0,1\}$, ciphertexts $\mathsf{ct}_1, \ldots, \mathsf{ct}_\nu$, it runs $\mathsf{ct}_f \leftarrow \mathsf{Eval}'(\mathsf{pk}, f, \omega, \mathsf{ct}_1, \ldots, \mathsf{ct}_\nu)$ with scaling factor $\omega = \lceil \log(B) \rceil + 1$, where the algorithm Eval' is described below, for the batch correctness property.

We demonstrate that the GSW FHE satisfies the batch correctness property.

Proposition 1 (Batch correctness)**.** *For all polynomials δ, the* GSW_δ *scheme described above satisfies batch correctness, as per Definition 3.1.*

Proof: We present the following PPT algorithm:

• $\underline{\mathsf{Eval}'(\mathsf{pk}, f, \omega, \mathsf{ct}_1, \ldots, \mathsf{ct}_\nu)}$:
Given the public pk, a depth-$\delta(\lambda)$ arithmetic circuit $f : \{0,1\}^\nu \to \mathbb{Z}_N$, a scaling factor $\omega < \log(N)$, ciphertexts $\mathsf{ct}_1, \ldots, \mathsf{ct}_\nu$, it evaluates the circuit gate by gate as follows.

- Addition gate between ct_i and ct_j: return $\mathsf{ct}_i + \mathsf{ct}_j \in \mathbb{Z}_N^{(\kappa+1)\times w}$.
- Multiplication gate between ct_i and ct_j: return $\mathsf{ct}_i \cdot \mathsf{BD}(\mathsf{ct}_j) \in \mathbb{Z}_N^{(\kappa+1)\times w}$, where $\mathsf{BD}(\mathsf{ct}_j) \in \{0,1\}^{w \times w}$ denotes the binary decomposition of $\mathsf{ct}_j \in \mathbb{Z}_N^{(\kappa+1)\times w}$.

By recursively applying the above operations, one can turn the ciphertexts $\mathsf{ct}_1, \ldots, \mathsf{ct}_\nu$ into $\mathbf{C}_f^i = \mathbf{UR}_f^i + f_i(\mathbf{m})\mathbf{G} \in \mathbb{Z}_N^{(\kappa+1)\times w}$, where $f_i(\mathbf{m}) \in \{0,1\}$ denotes the i'th bit of the binary decomposition of $f(\mathbf{m}) \in \mathbb{Z}_N$, that is, $f(\mathbf{m}) = \sum_{i=0}^{\lceil \log(N) \rceil - 1} 2^i f_i(\mathbf{m})$. For all $i \in [0, \lceil \log(N) \rceil - 1]$, we have $\|\mathbf{R}_f^i\|_\infty \leq (w+1)^\delta$. By definition of the matrix $\mathbf{G}$, choosing the $\kappa \cdot \lceil \log(N) \rceil + i + \omega + 1$'th column of $\mathbf{C}_f^i$ yields:

$$\mathbf{c}_f^i = \left(\mathbf{A}\mathbf{r}_f^i, (\mathbf{s}^\top \mathbf{A} + \mathbf{e}^\top)\mathbf{r}_f^i + 2^{\omega+i} f_i(\mathbf{m}) \right) \in \mathbb{Z}_N^{\kappa+1}.$$

Summing up for all $i \in [0, \lceil \log(N) \rceil - 1]$, we get: $\mathsf{ct}'_f = \left(\mathbf{A}\mathbf{r}_f, (\mathbf{s}^\top \mathbf{A} + \mathbf{e}^\top)\mathbf{r}_f + 2^\omega f(\mathbf{m}) \right) \in \mathbb{Z}_N^{\kappa+1}$, where $\mathbf{r}_f = \sum_{i=0}^{\lceil \log(N) \rceil - 1} \mathbf{r}_f^i$ of norm $\|\mathbf{r}_f\|_\infty \leq (w+1)^\delta \lceil \log(N) \rceil$. It outputs the evaluated ciphertext $\mathsf{ct}_f = \mathsf{BD}(\mathsf{ct}'_f) \in \{0,1\}^w$.

The evaluated ciphertext $\mathsf{ct}_f \in \{0,1\}^w$ is such that:

$$\mathsf{sk}^\top \mathsf{ct}_f = -\mathbf{s}^\top \mathbf{A}\mathbf{r}_f + (\mathbf{s}^\top \mathbf{A} + \mathbf{e}^\top)\mathbf{r}_f + 2^\omega f(\mathbf{m}) = 2^\omega f(\mathbf{m}) + \mathsf{noise}_f \in \mathbb{Z}_N,$$

where $\mathsf{noise}_f = \mathbf{e}^\top \mathbf{r}_f$. We have $|\mathsf{noise}_f| \leq (w+1)^\delta \lceil \log(N) \rceil B_\chi m = B$. □

We turn to proving that it also satisfies the randomness homomorphism property.

Proposition 2 (Randomness homomorphism)**.** *For all polynomials δ, the* GSW_δ *scheme satisfies the randomness homomorphism property as per Definition 3.2.*

Proof: We present the following PPT algorithms:

• $\underline{\mathsf{Enc}^\star(\mathsf{pk}, m; \mathbf{r}^\star)}$:

Given the public pk, a message $m \in \mathbb{Z}$, the randomness $\mathbf{r}^\star \in [-B^\star, B^\star]^m$, it computes $\mathsf{ct}' = \left(\mathbf{A}\mathbf{r}^\star, (\mathbf{s}^\top\mathbf{A} + \mathbf{e}^\top)\mathbf{r}^\star + m\right) \in \mathbb{Z}_N^{\kappa+1}$, and outputs $\mathsf{ct} = \mathsf{BD}(\mathsf{ct}') \in \{0,1\}^w$.

- $\underline{\mathsf{Eval}_{\mathsf{rand}}(\mathsf{pk}, f, (\mathbf{R}_i)_{i\in[\nu]}, (m_i)_{i\in[\nu]})}$:

This algorithm is similar to the ciphertext evaluation algorithm. Namely, given the public pk, a depth-$\delta(\lambda)$ arithmetic circuit $f : \{0,1\}^\nu \to \mathbb{Z}_N$, randomness $\mathbf{R}_1, \ldots, \mathbf{R}_\nu \in [-1,1]^{m\times w}$, it evaluates the circuit gate by gate as follows.

- Addition gate between $\mathbf{R}_i$ and $\mathbf{R}_j$: return $\mathbf{R}_i + \mathbf{R}_j \in \mathbb{Z}_N^{m\times w}$.
- Multiplication gate between $\mathbf{R}_i$ and $\mathbf{R}_j$: compute $\mathsf{ct}_j = \mathsf{Enc}_{\mathsf{pk}}(m_j; \mathbf{R}_j)$, return $\mathbf{R}_i\mathsf{BD}(\mathsf{ct}_j) + m_i\mathbf{R}_j \in \mathbb{Z}_N^{m\times w}$, where $\mathsf{BD}(\mathsf{ct}_j) \in \{0,1\}^{w\times w}$ denotes the binary decomposition of $\mathsf{ct}_j \in \mathbb{Z}_N^{(\kappa+1)\times w}$.

By recursively applying the above operations, one can turn the randomness $\mathbf{R}_1, \ldots, \mathbf{R}_n$ into $\mathbf{R}_f^i \in \mathbb{Z}_N^{m\times w}$ such that: $\mathbf{C}_f^i = \mathbf{U}\mathbf{R}_f^i + f_i(\mathbf{m})\mathbf{G} \in \mathbb{Z}_N^{(\kappa+1)\times w}$, for all $i \in [0, \lceil\log(N)\rceil - 1]$; and $\|\mathbf{R}_f^i\|_\infty \le (w+1)^\delta$. By definition of the matrix $\mathbf{G}$, choosing the $\kappa\lceil\log(N)\rceil + i + 1$'th column of $\mathbf{R}_f^i$ yields $\mathbf{r}_f^i \in \mathbb{Z}_N^m$ such that: $\mathbf{c}_f^i = \left(\mathbf{A}\mathbf{r}_f^i, (\mathbf{s}^\top\mathbf{A} + \mathbf{e}^\top)\mathbf{r}_f^i + 2^i f_i(\mathbf{m})\right) \in \mathbb{Z}_N^{\kappa+1}$, and $\|\mathbf{r}_f^i\|_\infty \le (w+1)^\delta$. Summing up for all $i \in [0, \lceil\log(N)\rceil - 1]$, we get: $\mathbf{r}_f = \sum_{i=0}^{\lceil\log(N)\rceil-1} \mathbf{r}_f^i \in \mathcal{R}^\star$ such that $\mathsf{ct}'_f = \left(\mathbf{A}\mathbf{r}_f, (\mathbf{s}^\top\mathbf{A} + \mathbf{e}^\top)\mathbf{r}_f + f(\mathbf{m})\right) \in \mathbb{Z}_N^{\kappa+1}$. It outputs the evaluated randomness $\mathbf{r}_f$. □

3.2.3 SRL Security

Before proving the SRL security of GSW under the LWE assumption, we describe new trapdoor generation and pre-image sampling algorithms that are inspired by those from [MP12]. As in prior works, the trapdoor generation algorithm generates a matrix $\mathbf{U} \in \mathbb{Z}_N^{d\times m}$ that is statistically close to uniformly random over $\mathbb{Z}_N^{d\times m}$, together with an associated trapdoor $T_{\mathbf{U}}$. The pre-image sampling algorithm, given a target vector $\mathbf{t} \in \mathbb{Z}_N^d$, produces a short pre-image, that is, a short vector $\mathbf{r} \in \mathbb{Z}_N^m$ such that $\mathbf{U}\mathbf{r} = \mathbf{t}$. In these works, the distribution of these short pre-images is independent of the trapdoor — typically they follow a discrete (spherical) Gaussian distribution. Our requirements are slightly different: a pre-image produced by our sampling algorithm when given as input a target vector $\mathbf{t} \in \mathbb{Z}_N^d$ should be statistically close to a pre-image produced by our sampling algorithm when given as input the vector $\mathbf{0} \in \mathbb{Z}_N^d$, shifted by a much smaller pre-image of $\mathbf{t}$. That is, if a very short pre-image is given, adding a somewhat short pre-image of $\mathbf{0}$ (produced by the sampling algorithm) to it will produce a pre-image that looks like a fresh output of the sampling algorithm on input $\mathbf{t}$. This inherently requires smudging size noises, which implies the use of an exponential-size modulus q. In fact this property is not known to hold for existing trapdoor generation and pre-image sampling algorithms using polynomial-size modulus.

We prove this property for the concrete algorithms provided in [MP12], which we simplify since we can afford to use smudging-size noises. We provide a self-contained description of the scheme and its proofs here.

Lattice trapdoors.

- $\underline{\mathsf{TrapGen}(1^\lambda, N, d)}$:

Given as input the security parameter $\lambda \in \mathbb{N}$, a modulus $N \in \mathbb{N}$, a dimension $d \in \mathbb{N}$, it sets

$\widetilde{m} = d\lceil\log(N)\rceil + 2\lambda$, $w = d\lceil\log(N)\rceil$, $m = \widetilde{m} + w$, computes the gadget matrix $\mathbf{G} = \mathbf{g}^\top \otimes \mathrm{Id} = \begin{pmatrix} \mathbf{g}^\top & 0 & \cdots \\ 0 & \mathbf{g}^\top & \\ \vdots & & \ddots \end{pmatrix} \in \mathbb{Z}_N^{d\times w}$ where $\mathrm{Id} \in \mathbb{Z}_N^{d\times d}$ denotes the identity matrix and $\mathbf{g} = (1, 2, \ldots, 2^{\lceil\log(N)\rceil-1}) \in \mathbb{Z}_N^{\lceil\log(N)\rceil}$, $\widetilde{\mathbf{U}} \leftarrow_{\mathsf{R}} \mathbb{Z}_N^{d\times\widetilde{m}}$, $\mathbf{R} \leftarrow_{\mathsf{R}} [-1,1]^{\widetilde{m}\times w}$, $\mathbf{U} = (\widetilde{\mathbf{U}} \| -\widetilde{\mathbf{U}}\mathbf{R}+\mathbf{G}) \in \mathbb{Z}_N^{d\times m}$, $T_{\mathbf{U}} = \mathbf{R}$. It outputs $(\mathbf{U}, T_{\mathbf{U}})$.

• $\underline{\mathsf{PreImSamp}(\mathbf{U}, T_{\mathbf{U}}, \mathbf{t}, B)}$:

Given as input the matrix $\mathbf{U}$, the trapdoor $T_{\mathbf{U}}$, a target vector $\mathbf{t} \in \mathbb{Z}_N^d$ and a bound $B \in \mathbb{N}$, it samples $\mathbf{v} \leftarrow_{\mathsf{R}} [-B, B]^m$, sets $\mathbf{b} = \mathsf{BD}(\mathbf{U}\mathbf{v} + \mathbf{t}) \in \{0,1\}^{w\times w}$ which denotes the binary decomposition of $\mathbf{U}\mathbf{v} + \mathbf{t} \in \mathbb{Z}_N^d$. It outputs $\begin{pmatrix} \mathbf{R}\mathbf{b} \\ \mathbf{b} \end{pmatrix} - \mathbf{v} \in \mathbb{Z}_N^m$.

We show the following properties hold.

Proposition 3 (Correctness of $\mathsf{TrapGen}$)**.** *For all λ, N, d, writing $m = 2d\lceil\log(N)\rceil + 2\lambda$, the following distributions have statistical distance at most $2^{-\lambda}$:*

$$\left\{\mathbf{U} \leftarrow_{\mathsf{R}} \mathbb{Z}_N^{d\times m} : \mathbf{U}\right\}$$
$$\left\{(\mathbf{U}, T_{\mathbf{U}}) \leftarrow \mathsf{TrapGen}(1^\lambda, N, d) : \mathbf{U}\right\}.$$

Proof: The proposition follows readily from Lemma 2.1 (leftover hash lemma). □

Proposition 4 (Correctness of $\mathsf{PreImSamp}$)**.** *For all $\lambda, q, d, B \in \mathbb{N}$, all $(\mathbf{U}, T_{\mathbf{U}})$ in the support of $\mathsf{TrapGen}(1^\lambda, N, d)$, all $\mathbf{t} \in \mathbb{Z}_N^d$, all $\mathbf{r} \in \mathbb{Z}_N^m$ in the support of $\mathsf{PreImSamp}(\mathbf{U}, T_{\mathbf{U}}, \mathbf{t}, B)$, are such $\mathbf{U}\mathbf{r} = \mathbf{t}$ and $\|\mathbf{r}\|_\infty < B + w$.*

Proof: Straightforward. □

Proposition 5 (Security)**.** *For all $\lambda, N, d, B \in \mathbb{N}$, writing $m = 2d\lceil\log(N)\rceil + 2\lambda$, for all $\mathbf{w} \in \mathbb{Z}_N^m$ such that $\|\mathbf{w}\|_\infty < B'$, the statistical distance of the two following distributions is upper-bounded by B'/B:*

$$\{(\mathbf{U}, T_{\mathbf{U}}) \leftarrow \mathsf{TrapGen}(1^\lambda, N, d), \widetilde{\mathbf{r}_0} \leftarrow_{\mathsf{R}} \mathsf{PreImSamp}(\mathbf{U}, T_{\mathbf{U}}, \mathbf{0}, B) : \widetilde{\mathbf{r}_0} + \mathbf{w} \in \mathbb{Z}_N^m\}$$
$$\{(\mathbf{U}, T_{\mathbf{U}}) \leftarrow \mathsf{TrapGen}(1^\lambda, N, d), \widetilde{\mathbf{r}} \leftarrow \mathsf{PreImSamp}(\mathbf{U}, T_{\mathbf{U}}, \mathbf{U}\mathbf{w}, B) : \widetilde{\mathbf{r}}\}$$

Proof: By definition of $\mathsf{PreImSamp}$ we have: $\widetilde{\mathbf{r}_0} = \begin{pmatrix} \mathbf{R}\mathbf{b} \\ \mathbf{b} \end{pmatrix} - \mathbf{v}$ where $\mathbf{v} \leftarrow_{\mathsf{R}} [-B, B]^m$ and $\mathbf{b} = \mathsf{BD}(\mathbf{U}\mathbf{v}) \in \{0,1\}^w$. For all $\mathbf{w} \in \mathbb{Z}_N^m$ such that $\|\mathbf{w}\|_\infty < B'$, by Lemma 2.2 (smudging), the following distributions have statistical distance B'/B: $\{\mathbf{v} \leftarrow_{\mathsf{R}} [-B, B]^m : \mathbf{v}\}$ and $\{\mathbf{v} \leftarrow_{\mathsf{R}} [-B, B]^m : \mathbf{v} + \mathbf{w}\}$. This implies that $\widetilde{\mathbf{r}_0} + \mathbf{w} \approx \begin{pmatrix} \mathbf{R}\mathbf{b}' \\ \mathbf{b}' \end{pmatrix} - \mathbf{v}$, where $\mathbf{b}' = \mathsf{BD}(\mathbf{U}\mathbf{v} + \mathbf{U}\mathbf{w})$. The latter is identically distributed to $\mathsf{PreImSamp}(\mathbf{U}, T_{\mathbf{U}}, \mathbf{U}\mathbf{w}, B)$. □

Theorem 3.5 (SRL security)**.** *Assume the (subexponential) security of the LWE assumption holds. Then, for all polynomials δ, GSW_δ is (subexponentially) SRL secure.*

Proof: For all nuPPT adversaries $\mathcal{A}$, all $\lambda \in \mathbb{N}$, we use the following hybrid experiments.

• $\underline{\mathcal{H}^0_{\lambda,\mathcal{A}}}$: is the experiment $\mathsf{Exp}^{\mathrm{GSW}_\delta}_{\lambda,\mathcal{A}}$ from Definition 2.8.

• $\mathcal{H}^1_{\lambda,\mathcal{A}}$: is as $\mathcal{H}^0_{\lambda,\mathcal{A}}$ except the LWE sample $\mathbf{s}^\top\mathbf{A}+\mathbf{e}^\top$ from the public key is switched to a uniformly random vector using the LWE assumption. That is, the public key is computed as follows: $\mathbf{A} \leftarrow_{\mathsf{R}} \mathbb{Z}_N^{\kappa\times m}$, $\mathbf{v} \leftarrow_{\mathsf{R}} \mathbb{Z}_N^m$, $\mathbf{U} = \begin{pmatrix}\mathbf{A}\\ \mathbf{v}^\top\end{pmatrix}$; the gadget matrix $\mathbf{G}$ is computed as in $\mathcal{H}^0_{\lambda,\mathcal{A}}$, and $\mathsf{pk} = (B, \mathbf{U}, \mathbf{G})$. The secret key is also computed as in $\mathcal{H}^0_{\lambda,\mathcal{A}}$ (but now it is uncorrelated with pk), namely: $\mathbf{s} \leftarrow_{\mathsf{R}} \chi^\kappa$, $\mathsf{sk} = (-\mathbf{s}, 1) \otimes \mathbf{g} \in \mathbb{Z}_N^w$. The challenge ciphertext is computed as $\mathsf{ct} = \mathsf{Enc}_{\mathsf{pk}}(\mathbf{m}^\star; \mathbf{r})$ and the oracle $\mathcal{O}_{\mathsf{SRL}}(\mathbf{m}^\star, \mathbf{r})$ behaves as in $\mathcal{H}^0_{\lambda,\mathcal{A}}$. The LWE assumption implies that for all nuPPT $\mathcal{A}$, there exists a negligible function μ such that for all $\lambda \in \mathbb{N}$, $|\Pr[\mathcal{H}^0_{\lambda,\mathcal{A}} = 1] - \Pr[\mathcal{H}^1_{\lambda,\mathcal{A}} = 1]| \le \mu(\lambda)$, since these experiments can be efficiently simulated from $\mathbf{U}$ and the winning condition can be efficiently checked.

• $\mathcal{H}^2_{\lambda,\mathcal{A}}$: is as $\mathcal{H}^1_{\lambda,\mathcal{A}}$ except the matrix $\mathbf{U}$ from the public key is sampled from $(\mathbf{U}, T_{\mathbf{U}}) \leftarrow \mathsf{TrapGen}(1^\lambda, N, \kappa+1)$. By Property 3, this is statistically close (within statistical distance $2^{-\Omega(\lambda)}$) to generating a uniformly random $\mathbf{U} \leftarrow_{\mathsf{R}} \mathbb{Z}_N^{(\kappa+1)\times m}$ as done in $\mathcal{H}^1_{\lambda,\mathcal{A}}$. The experiments can be simulated from $\mathbf{U}$, thus, for all nuPPT $\mathcal{A}$, we have:

$$\{\mathcal{H}^1_{\lambda,\mathcal{A}}\}_{\lambda\in\mathbb{N}} \approx_s \{\mathcal{H}^2_{\lambda,\mathcal{A}}\}_{\lambda\in\mathbb{N}}.$$

• $\mathcal{H}^3_{\lambda,\mathcal{A}}$: is as $\mathcal{H}^2_{\lambda,\mathcal{A}}$ except we use the oracle $\widetilde{\mathcal{O}}_{\mathsf{SRL}}$ instead of $\mathcal{O}_{\mathsf{SRL}}$:

> $\widetilde{\mathcal{O}}_{\mathsf{SRL}}(\mathbf{m}^\star, \mathsf{ct})$:
> $\mathbf{r}^\star \leftarrow_{\mathsf{R}} \mathcal{R}^\star$, $\mathsf{ct}^\star = \mathsf{Enc}^\star_{\mathsf{pk}}(0; \mathbf{r}^\star)$
> $(f, \alpha) \leftarrow \mathcal{A}(\mathsf{ct}^\star)$
> $\mathsf{BD}(\mathsf{ct}'_f) = \mathsf{Eval}'(\mathsf{pk}, f, 0, \mathsf{ct})$. Parse $\mathsf{ct}'_f = \big(\mathbf{A}\mathbf{r}_f, \mathbf{v}^\top\mathbf{r}_f + f(\mathbf{m}^\star)\big) \in \mathbb{Z}_N^{\kappa+1}$.
> Compute $\mathbf{t}_f = (\mathbf{A}\mathbf{r}_f, \mathbf{v}^\top\mathbf{r}_f) \in \mathbb{Z}_N^{\kappa+1}$, and $\widetilde{\mathbf{r}}_f \leftarrow \mathsf{PreImSamp}(\mathbf{U}, T_{\mathbf{U}}, \mathbf{t}_f, B^\star 2^{-\lambda/2})$.
> If $f(\mathbf{m}^\star) = \alpha$, and f is of depth δ, then $\mathsf{leak} = \mathbf{r}^\star - \widetilde{\mathbf{r}}_f \in \mathcal{R}^\star$.
> Otherwise, $\mathsf{leak} = \perp$. Return leak.

Note that the oracle $\widetilde{\mathcal{O}}_{\mathsf{SRL}}$ only takes as input the message $\mathbf{m}^\star \in \{0,1\}^*$ and the challenge ciphertext $\mathsf{ct} = \mathsf{Enc}_{\mathsf{pk}}(\mathbf{m}^\star; \mathbf{r})$, but not the randomness $\mathbf{r}$ itself. Instead of computing the evaluated randomness $\mathbf{r}_f = \mathsf{Eval}_{\mathsf{rand}}(\mathsf{pk}, f, \mathbf{m}^\star, \mathbf{r})$, it computes a small $\widetilde{\mathbf{r}}_f$ that is consistent with the evaluated ciphertext $\mathsf{ct}_f = \mathsf{BD}(\mathsf{ct}'_f)$, that is, such that $\mathsf{ct}'_f = (\mathbf{A}\widetilde{\mathbf{r}}_f, \mathbf{v}^\top\widetilde{\mathbf{r}}_f + f(\mathbf{m}))$. Clearly, the distributions: $(\mathsf{ct}, \mathbf{r}_f)$, which corresponds to $\mathcal{H}^2_{\lambda,\mathcal{A}}$ and $(\mathsf{ct}, \widetilde{\mathbf{r}}_f)$, which corresponds to $\mathcal{H}^3_{\lambda,\mathcal{A}}$ are distinct — for one thing, the first distribution has less entropy than the second distribution where $\widetilde{\mathbf{r}}_f$ is sampled freshly. However, the value $\widetilde{\mathbf{r}}_f$ is shielded by the noisy randomness $\mathbf{r}^\star \leftarrow_{\mathsf{R}} \mathcal{R}^\star$. Because it is of much larger magnitude than $\mathbf{r}_f$ and $\widetilde{\mathbf{r}}_f$, the latter can smudge the difference $\delta_f = \mathbf{r}_f - \widetilde{\mathbf{r}}_f$, which would successfully transition from $\mathcal{H}^2_{\lambda,\mathcal{A}}$ to $\mathcal{H}^3_{\lambda,\mathcal{A}}$. To effectively hide δ_f, we need to make sure $\mathbf{r}^\star \in \mathbb{Z}^m$ itself is hidden. Partial information is revealed in $\mathsf{ct}^\star = \mathsf{Enc}^\star_{\mathsf{pk}}(0; \mathbf{r}^\star)$, of the form $\mathbf{U}\mathbf{r}^\star \in \mathbb{Z}_N^{\kappa+1}$. Intuitively, the component of $\mathbf{r}^\star$ along $\mathbf{U}$ is revealed by $\mathsf{ct}^\star$, but the remaining entropy of $\mathbf{r}^\star$ is hidden; in particular, its component along $\mathbf{U}^\perp$, the orthogonal space of $\mathbf{U}$, is hidden. Because $\widetilde{\mathbf{r}}_f$ is consistent with ct_f, we have $\mathbf{U}\delta_f = \mathbf{0}$; that is, δ_f is orthogonal to $\mathbf{U}$. The orthogonal component of $\mathbf{r}^\star$ can simply smudge δ_f. This argument is formalized in Lemma 3.1. Overall, for all nuPPT $\mathcal{A}$, we have:

$$\{\mathcal{H}^2_{\lambda,\mathcal{A}}\}_{\lambda\in\mathbb{N}} \approx_s \{\mathcal{H}^3_{\lambda,\mathcal{A}}\}_{\lambda\in\mathbb{N}}.$$

To complete the proof of Theorem 3.5, we now show that for all nuPPT $\mathcal{A}$, all $\lambda \in \mathbb{N}$, we have: $\Pr[\mathcal{H}^3_{\lambda,\mathcal{A}} = 1] \le 1/2$. To do so, we consider the event fail (and the complementary event $\overline{\mathsf{fail}}$), which

happens when $\mathcal{A}$ chooses a pair of messages $(\mathbf{m}^0, \mathbf{m}^1)$ and makes a query of the form $(f^\star, \alpha^\star)$ to $\mathcal{O}_{\mathsf{SRL}}$ such that $f^\star(\mathbf{m}^0) \neq f^\star(\mathbf{m}^1)$.

First, we show that $\Pr[\mathcal{H}^3_{\lambda,\mathcal{A}} = 1|\mathsf{fail}] \leq 1/2$. This follows from the fact that conditioning on fail, the query $(f^\star, \alpha^\star)$ makes $\mathcal{O}_{\mathsf{SRL}}$ output $\perp$ with probability $1/2$ over the choice of $b \leftarrow_{\mathsf{R}} \{0,1\}$, in which case the experiment $\mathcal{H}^3_{\lambda,\mathcal{A}}$ outputs 0.

Finally, we show that $\Pr[\mathcal{H}^3_{\lambda,\mathcal{A}} = 1|\overline{\mathsf{fail}}] \leq 1/2$. This follows from the fact that in the experiment $\mathcal{H}^3_{\lambda,\mathcal{A}}$, the only information revealed about the random bit b is $f(\mathbf{m}^\star)$ where $\mathbf{m}^\star = \mathbf{m}^b$. Since we condition on $\overline{\mathsf{fail}}$, we know that $f(\mathbf{m}^0) = f(\mathbf{m}^1)$. Thus, there is no information revealed about b and $\Pr[\mathcal{H}^3_{\lambda,\mathcal{A}} = 1|\overline{\mathsf{fail}}] = 1/2$. □

Lemma 3.1. *For all nuPPT $\mathcal{A}$, we have: $\{\mathcal{H}^2_{\lambda,\mathcal{A}}\}_{\lambda\in\mathbb{N}} \approx_s \{\mathcal{H}^3_{\lambda,\mathcal{A}}\}_{\lambda\in\mathbb{N}}$.*

Proof: We introduce intermediate hybrids $\mathcal{H}^{2.i}_{\lambda,\mathcal{A}}$ for $i = 1, 2$ which are defined for all nuPPT adversaries $\mathcal{A}$ and all $\lambda \in \mathbb{N}$ as follows.

The experiment $\mathcal{H}^{2.1}_{\lambda,\mathcal{A}}$ is as $\mathcal{H}^2_{\lambda,\mathcal{A}}$ except is uses the following oracle $\mathcal{O}^1_{\mathsf{SRL}}$ instead of $\mathcal{O}_{\mathsf{SRL}}$.

$\underline{\mathcal{O}^1_{\mathsf{SRL}}(\mathbf{m}^\star, \mathbf{r})}$:
$\mathbf{r}^\star \leftarrow_{\mathsf{R}} \mathcal{R}^\star$, $\widetilde{\mathbf{r}_0} \leftarrow \mathsf{PreImSamp}(\mathbf{U}, T_{\mathbf{U}}, \mathbf{0}, B^\star 2^{-\lambda/2})$, $\mathsf{ct}^\star = \mathsf{Enc}^\star_{\mathsf{pk}}(0; \mathbf{r}^\star - \widetilde{\mathbf{r}_0})$
$(f, \alpha) \leftarrow \mathcal{A}(\mathsf{ct}^\star)$
$\mathbf{r}_f = \mathsf{Eval}_{\mathsf{rand}}(\mathsf{pk}, f, \mathbf{r}, \mathbf{m}^\star)$.
If $f(\mathbf{m}^\star) = \alpha$ and f is of depth δ, then $\mathsf{leak} = \mathbf{r}^\star - \widetilde{\mathbf{r}_0} - \mathbf{r}_f \in \mathcal{R}^\star$.
Otherwise, $\mathsf{leak} = \perp$. Return leak.

We first prove that for all nuPPT $\mathcal{A}$, we have:

$$\{\mathcal{H}^2_{\lambda,\mathcal{A}}\}_{\lambda\in\mathbb{N}} \approx_s \{\mathcal{H}^{2.1}_{\lambda,\mathcal{A}}\}_{\lambda\in\mathbb{N}}.$$

The only difference between these experiments is that $\mathcal{O}^1_{\mathsf{SRL}}$ subtract a pre-image of $\mathbf{0} \in \mathbb{Z}^m_N$ from the shield, that is, it uses $\mathbf{r}^\star - \widetilde{\mathbf{r}_0}$ with $\mathbf{r}^\star \leftarrow_{\mathsf{R}} \mathcal{R}^\star$ and $\widetilde{\mathbf{r}_0} \leftarrow \mathsf{PreImSamp}(\mathbf{U}, T_{\mathbf{U}}, \mathbf{0}, B^\star 2^{-\lambda/2})$ instead of $\mathbf{r}^\star$.

By Property 4, $\widetilde{\mathbf{r}_0} \in \mathbb{Z}^m_N$ is such that $\|\widetilde{\mathbf{r}_0}\|_\infty < B^\star 2^{\lambda/2}$. Thus, by Lemma 2.2 (smudging), the following distributions have statistical distance at most $2^{-\lambda/2}$:

$$\{\mathbf{r}^\star \leftarrow_{\mathsf{R}} [-B^\star, B^\star]^m : \mathbf{r}^\star\} \quad \text{and} \quad \{\mathbf{r}^\star \leftarrow_{\mathsf{R}} [-B^\star, B^\star]^m : \mathbf{r}^\star - \widetilde{\mathbf{r}_0}\}.$$

The leftmost distribution corresponds to the experiment $\mathcal{H}^2_{\lambda,\mathcal{A}}$ (with post-processing), whereas the rightmost distribution corresponds to the experiment $\mathcal{H}^{2.1}_{\lambda,\mathcal{A}}$ (with the same post-processing). This completes the proof that $\{\mathcal{H}^2_{\lambda,\mathcal{A}}\}_{\lambda\in\mathbb{N}} \approx_s \{\mathcal{H}^{2.1}_{\lambda,\mathcal{A}}\}_{\lambda\in\mathbb{N}}$.

Now, we introduce another intermediate hybrid, $\mathcal{H}^{2.2}_{\lambda,\mathcal{A}}$, which uses the oracle $\mathcal{O}^2_{\mathsf{SRL}}$ instead of $\mathcal{O}^1_{\mathsf{SRL}}$, where $\mathcal{O}^2_{\mathsf{SRL}}$ behaves just as $\mathcal{O}^1_{\mathsf{SRL}}$ with the only exception that $\mathsf{ct}^\star$ is encrypted using the randomness $\mathbf{r}^\star$ (as opposed to randomness $\mathbf{r}^\star - \widetilde{\mathbf{r}_0}$):

$\underline{\mathcal{O}^2_{\mathsf{SRL}}(\mathbf{m}^\star, \mathbf{r})}$:
$\mathbf{r}^\star \leftarrow_{\mathsf{R}} \mathcal{R}^\star$, $\mathsf{ct}^\star = \mathsf{Enc}^\star_{\mathsf{pk}}(0; \mathbf{r}^\star)$
$(f, \alpha) \leftarrow \mathcal{A}(\mathsf{ct}^\star)$
$\mathbf{r}_f = \mathsf{Eval}_{\mathsf{rand}}(\mathsf{pk}, f, \mathbf{r}, \mathbf{m}^\star)$, $\widetilde{\mathbf{r}_0} \leftarrow \mathsf{PreImSamp}(\mathbf{U}, T_{\mathbf{U}}, \mathbf{0}, B^\star 2^{-\lambda/2})$.
If $f(\mathbf{m}^\star) = \alpha$ and f is of depth δ, then $\mathsf{leak} = \mathbf{r}^\star - \widetilde{\mathbf{r}_0} - \mathbf{r}_f \in \mathcal{R}^\star$.
Otherwise, $\mathsf{leak} = \perp$. Return leak.

By Property 4, we have, $\mathbf{U}\widetilde{\mathbf{r}_0} = \mathbf{0}$. This implies $\mathsf{Enc}^\star_{\mathsf{pk}}(0; \mathbf{r}^\star - \widetilde{\mathbf{r}_0}) = \mathsf{Enc}^\star_{\mathsf{pk}}(0; \mathbf{r}^\star)$. Thus, for all nuPPT $\mathcal{A}$, we have:

$$\{\mathcal{H}^{2.1}_{\lambda,\mathcal{A}}\}_{\lambda\in\mathbb{N}} = \{\mathcal{H}^{2.2}_{\lambda,\mathcal{A}}\}_{\lambda\in\mathbb{N}}.$$

To conclude the proof of this lemma, we now prove that for all nuPPT $\mathcal{A}$, we have:

$$\{\mathcal{H}^{2.2}_{\lambda,\mathcal{A}}\}_{\lambda\in\mathbb{N}} \approx_s \{\mathcal{H}^{3}_{\lambda,\mathcal{A}}\}_{\lambda\in\mathbb{N}}.$$

To do so, we note that $\mathbf{r}_f \in \mathbb{Z}_N^m$ is such that $\|\mathbf{r}_f\|_\infty < B^\star 2^{-\lambda}$. Moreover, it is independent of the vector $\widetilde{\mathbf{r}_0} \leftarrow \mathsf{PreImSamp}(\mathbf{U}, T_{\mathbf{U}}, \mathbf{0}, B^\star 2^{-\lambda/2})$. Therefore, we can use Proposition 5, which states that for all vectors $\mathbf{r}_f \in \mathbb{Z}_N^m$ such that $\|\mathbf{r}_f\|_\infty < B^\star 2^{-\lambda}$, the following distributions have statistical distance at most $2^{-\lambda/2}$:

$$\{\widetilde{\mathbf{r}_0} \leftarrow_{\mathsf{R}} \mathsf{PreImSamp}(\mathbf{U}, T_{\mathbf{U}}, \mathbf{0}, B^\star 2^{-\lambda/2}) : \widetilde{\mathbf{r}_0} + \mathbf{r}_f \in \mathbb{Z}_N^m\} \text{ and } \{\widetilde{\mathbf{r}_f} \leftarrow \mathsf{PreImSamp}(\mathbf{U}, T_{\mathbf{U}}, \mathbf{U}\mathbf{r}_f, B^\star 2^{-\lambda/2}) : \widetilde{\mathbf{r}_f}\}.$$

The leftmost distribution corresponds to the experiment $\mathcal{H}^{2.2}_{\lambda,\mathcal{A}}$ (with pre and post-processing), whereas the rightmost distribution corresponds to the experiment $\mathcal{H}^{3}_{\lambda,\mathcal{A}}$ (with the same pre and post-processing). □

4 Hintable Linearly Homomorphic Encryption

BDGM [BDGM20a] introduced the notion of "hintable" Linearly Homomorphic Encryption (LHE). Roughly speaking, an LHE scheme is said to be hintable if there is a secret-key algorithm that given a ciphertext, produces a "short" decryption hint. The latter can be used to decrypt the ciphertext is was generated from, without the secret key. It is also possible to generate a hint from a ciphertext only knowing the random coins used to produce that ciphertext (but without knowledge of the secret key), and the hints generated in these two ways should be statistically close. For our purposes (and as explained in the introduction), we will need to consider a notion of a hintable LHE satisfying a "weak circuit privacy" notion.

Additionally, we here generalize the notion of a hintable LHE to also consider "packed" LHE, where we can encrypt a vector of messages. Additionally, (just as we did for FHE), we will consider LHE with two encryption modes: a "normal" and an "extra noisy" mode. Linear functions can be evaluated on normal encryptions; furthermore *one* addition with a noisy encryption can be performed. More additions with noisy encryptions would lead to ill-formed ciphertexts that cannot be decrypted properly. (We introduce these extra generalizations to be able to obtain an instantiation based on LWE; these extra generalizations are not needed to capture DJ).

More precisely, we consider the notion of an $(\ell_1, \ell_2, h, \alpha)$-hintable packed LHE which enables operating over a plaintext space of length $\ell_2(\lambda)$ vectors over $\mathbb{Z}_N$ for some modulus $|N| \geq \ell_1(\lambda)$, and release hints of size $h(\lambda)$. We will be interested in schemes where either ℓ_1 or ℓ_2 can be made arbitrarily big, while keeping h the same (i.e, the hint will become significantly shorter than a single group element, or it will be significantly smaller than the packing capacity). We consider LHE schemes where decryption only recovers the encrypted message approximately, with some extra small noise. The parameter α quantifies the noise magnitude.

We will present two constructions satifying the notion of a hintable packed LHE. The first one is the Damgård-Jurik [DJ01] encryption scheme which is proven secure under the DCR assumption: this construction considers the setting where $\ell_2(\lambda) = 1$ (i.e., there is no packing, and instead the group elements are directly much larger than the size of the hint), and $\alpha(\lambda) = 0$, i.e. the decryption recovers the encrypted message perfectly. The second construction will instead be a tweaked version of Packed-Regev [Reg05, PVW08] which is secure under the LWE assumption; in this construction,

$\ell_1(\lambda)$ is small (comparable to the hint size), but instead $\ell_2(\lambda)$ can be made arbitrarily large (i.e., we can pack a large number of elements into a ciphertext and still keep the hint size small). The approximation parameter $\alpha(\lambda) = 2^{\lambda+1}$ (note that we will use a modulus N of exponential size in λ for this scheme).

4.1 Definition of Hintable LHE

We proceed to the formal defnition.

Definition 4.1 (hintable LHE)**.** *For any polynomial $\ell_1, \ell_2, h, \alpha$, an $(\ell_1, \ell_2, h, \alpha)$-*Hintable Packed LHE *comprises the following PPT algorithms:*

- $\mathsf{CRSgen}(1^\lambda)$*: given as input the security parameter $\lambda \in \mathbb{N}$, it outputs* crs*.*
- $\mathsf{Gen}(\mathsf{crs})$*: given as input* crs*, it outputs the tuple* $(\mathsf{pk}, \mathsf{sk}, \mathsf{td})$*, where* td *a trapdoor that will be used to compute decryption hints. This tuple defines the message space $\mathbb{Z}_N^{\ell_2(\lambda)}$, where $N \geq 2^{\ell_1(\lambda)}$.*
- $\mathsf{Enc}_{\mathsf{pk}}(\mathbf{x})$*: given as input the public key* pk *and a vector in $\mathbf{x} \in \mathbb{Z}_N^\nu$, it outputs a ciphertext* ct*.*
- $\mathsf{Enc}^\star_{\mathsf{pk}}(\mathbf{m})$*: given as input the public key* pk *and a message in $\mathbf{m} \in \mathbb{Z}_N^{\ell_2(\lambda)}$, it outputs a noisy ciphertext* $\mathsf{ct}^\star$*.*
- $\mathsf{Eval}(\mathsf{pk}, \mathsf{ct}, \mathsf{ct}^\star, \mathbf{y})$*: given as input the public key* pk*, ciphertexts* ct*, a noisy ciphertext* $\mathsf{ct}^\star$ *and a function $\mathbf{y} \in [-1, 1]^{\nu\ell_2}$, it outputs an evaluated ciphertext* $\mathsf{ct}_{\mathbf{y}}$*.*
- $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}^\star)$*: given as input the secret key and a (noisy or evaluated) ciphertext* $\mathsf{ct}^\star$*, it outputs a plaintext.*
- $\mathsf{SecHint}(\mathsf{td}, \mathsf{ct}^\star)$*: given as input the secret trapdoor* td *and a (noisy or evaluated) ciphertext* $\mathsf{ct}^\star$*, it outputs a decryption hint ρ.*
- $\mathsf{PubHint}(\mathsf{pk}, r)$*: given as input the public key and some random coins $r \in \mathcal{R}^\star$, where $\mathcal{R}^\star$ denotes the randomness space of* $\mathsf{Enc}^\star$*, it outputs a hint ρ.*
- $\mathsf{Rec}(\mathsf{pk}, \mathsf{ct}^\star, \rho)$*: given as input a (noisy or evaluated) ciphertext and a decryption hint ρ, it outputs a plaintext.*

These PPT algorithms additionally need to satisfy the properties listed below.

Property 4.1 (α-approximate correctness)**.** *For all $\lambda \in \mathbb{N}$, all* crs *in the support of* $\mathsf{CRSgen}(1^\lambda)$*, all tuples* $(\mathsf{pk}, \mathsf{sk}, \mathsf{td})$ *in the support of* $\mathsf{Gen}(\mathsf{crs})$ *that define the message space $\mathbb{Z}_N^{\ell_2(\lambda)}$ where $N \geq 2^{\ell_1(\lambda)}$, all messages $\mathbf{m} \in \mathbb{Z}_N^{\ell_2(\lambda)}$, we have:* $\Pr\left[\mathsf{ct}^\star \leftarrow \mathsf{Enc}^\star_{\mathsf{pk}}(\mathbf{m}), \mathbf{m}' = \mathsf{Dec}_{\mathsf{sk}}(\mathsf{ct}^\star) : \|\mathbf{m}' - \mathbf{m}\|_\infty < 2^{\alpha(\lambda)}\right] \in 1 - 2^{-\Omega(\lambda)}$.

Property 4.2 (Linear Homomorphism)**.** *For all polynomials $\nu(\cdot)$, all $\lambda \in \mathbb{N}$, all* crs *in the support of* $\mathsf{CRSgen}(1^\lambda)$*, all tuples* $(\mathsf{pk}, \mathsf{sk}, \mathsf{td})$ *in the support of* $\mathsf{Gen}(\mathsf{crs})$ *that define the message space $\mathbb{Z}_N^{\ell_2(\lambda)}$, all vectors $\mathbf{x} \in \mathbb{Z}_N^{\nu(\lambda)}$, all ciphertexts* ct *in the support of* $\mathsf{Enc}_{\mathsf{pk}}(\mathbf{x})$*, all messages $\mathbf{m} \in \mathbb{Z}_N^{\ell_2(\lambda)}$, all ciphertexts* $\mathsf{ct}^\star$ *in the support of* $\mathsf{Enc}^\star_{\mathsf{pk}}(\mathbf{m})$*, all vectors $\mathbf{y} = (\mathbf{y}_1, \ldots, \mathbf{y}_{\ell_2}) \in \{0,1\}^{\nu(\lambda)\ell_2}$, we have:* $\mathsf{Eval}(\mathsf{pk}, \mathsf{ct}, \mathsf{ct}^\star, \mathbf{y})$ *deterministically outputs a ciphertext in the support of* $\mathsf{Enc}^\star_{\mathsf{pk}}(m_1 + \mathbf{x}^\top \mathbf{y}_1, \ldots, m_{\ell_2} + \mathbf{x}^\top \mathbf{y}_{\ell_2})$.

Property 4.3 (α-approximate correctness of the secret hints)**.** *For all $\lambda \in \mathbb{N}$, all* crs *in the support of* $\mathsf{CRSgen}(1^\lambda)$*, all tuples* $(\mathsf{pk}, \mathsf{sk}, \mathsf{td})$ *in the support of* $\mathsf{Gen}(\mathsf{crs})$ *that define the message space* $\mathbb{Z}_N^{\ell_2(\lambda)}$*, for all messages* $\mathbf{m} \in \mathbb{Z}_N^{\ell_2(\lambda)}$*, we have:* $\Pr\left[\mathsf{ct}^\star \leftarrow \mathsf{Enc}^\star_{\mathsf{pk}}(\mathbf{m}), \rho \leftarrow \mathsf{SecHint}(\mathsf{sk}, \mathsf{ct}^\star), \mathbf{m}' = \mathsf{Rec}(\mathsf{pk}, \mathsf{ct}^\star, \rho) : \|\mathbf{m}' - \mathbf{m}\|_\infty \leq 2^{\alpha(\lambda)}\right] \in 1 - 2^{-\Omega(\lambda)}$.

Property 4.4 (Equivalence between public and secret hints)**.** *For all $\lambda \in \mathbb{N}$, all* crs *in the support of* $\mathsf{CRSgen}(1^\lambda)$*, all messages* $\mathbf{m} \in \mathbb{Z}_N^{\ell_2(\lambda)}$*, the following distributions have statistical distance at most* $2^{-\Omega(\lambda)}$*:*

$$\{(\mathsf{pk}, \mathsf{sk}, \mathsf{td}) \leftarrow \mathsf{Gen}(1^\lambda), \mathsf{ct}^\star \leftarrow \mathsf{Enc}^\star_{\mathsf{pk}}(\mathbf{m}), \rho \leftarrow \mathsf{SecHint}(\mathsf{sk}, \mathsf{ct}) : (\mathsf{pk}, \mathsf{ct}^\star, \rho)\}$$
$$\{(\mathsf{pk}, \mathsf{sk}, \mathsf{td}) \leftarrow \mathsf{Gen}(1^\lambda), r \leftarrow_{\mathsf{R}} \mathcal{R}^\star, \rho \leftarrow \mathsf{PubHint}(\mathsf{pk}, r), \mathsf{ct}^\star = \mathsf{Enc}^\star_{\mathsf{pk}}(\mathbf{m}; r) : (\mathsf{pk}, \mathsf{ct}^\star, \rho)\}$$

Property 4.5 (h-succinctness of hints)**.** *For all $\lambda \in \mathbb{N}$, all* crs *in the support of* $\mathsf{CRSgen}(1^\lambda)$*, all tuples* $(\mathsf{pk}, \mathsf{sk}, \mathsf{td})$ *in the support of* $\mathsf{Gen}(\mathsf{crs})$ *that define the message space* $\mathbb{Z}_N^{\ell_2(\lambda)}$*, all messages* $\mathbf{x} \in \mathbb{Z}_N^{\ell_2(\lambda)}$*, all ciphertexts* $\mathsf{ct}^\star$ *in the support of* $\mathsf{Enc}^\star_{\mathsf{pk}}(\mathbf{x})$*, all hints ρ in the support of* $\mathsf{SecHint}(\mathsf{sk}, \mathsf{ct}^\star)$ *are of size at most $h(\lambda)$.*

Property 4.6 (Weak circuit privacy)**.** *For all polynomials $\nu(\cdot)$, all $\lambda \in \mathbb{N}$, all* crs *in the support of* $\mathsf{CRSgen}(1^\lambda)$*, all tuples* $(\mathsf{pk}, \mathsf{sk}, \mathsf{td})$ *in the support of* $\mathsf{Gen}(\mathsf{crs})$ *that define the message space* $\mathbb{Z}_N^{\ell_2(\lambda)}$*, all messages* $\mathbf{m} \in \mathbb{Z}_N^{\ell_2}$*, all vectors* $\mathbf{x} \in \mathbb{Z}_N^{\nu(\lambda)}$*, all vectors* $\mathbf{y} \in [-1, 1]^{\nu(\lambda)\ell_2(\lambda)}$*, the following distributions have statistical distance at most* $2^{-\Omega(\lambda)}$*:*

$$\{\mathsf{ct} \leftarrow \mathsf{Enc}_{\mathsf{pk}}(\mathbf{x}), \mathsf{ct}^\star \leftarrow \mathsf{Enc}^\star_{\mathsf{pk}}(\mathbf{m}), \mathsf{ct}_{\mathbf{y}} = \mathsf{Eval}(\mathsf{pk}, \mathsf{ct}, \mathsf{ct}^\star, \mathbf{y}) : (\mathsf{pk}, \mathsf{crs}, \mathsf{ct}, \mathsf{ct}^\star, \mathsf{ct}_{\mathbf{y}})\}$$
$$\{\mathsf{ct} \leftarrow \mathsf{Enc}_{\mathsf{pk}}(\mathbf{x}), \mathsf{ct}_{\mathbf{y}} \leftarrow \mathsf{Enc}^\star_{\mathsf{pk}}(\mu), \mathsf{ct}^\star = \mathsf{Eval}(\mathsf{pk}, \mathsf{ct}, \mathsf{ct}_{\mathbf{y}}, -\mathbf{y}) : (\mathsf{crs}, \mathsf{pk}, \mathsf{ct}, \mathsf{ct}^\star, \mathsf{ct}_{\mathbf{y}})\},$$

where $\mu = (m_1 + \mathbf{x}^\top \mathbf{y}_1, \ldots, m_{\ell_2} + \mathbf{x}^\top \mathbf{y}_{\ell_2}) \in \mathbb{Z}_N^{\ell_2}$.

Property 4.7 (Density of the noisy ciphertexts)**.** *There exists a polynomial $s(\cdot)$ and a poly-time determinstic function* CTsample *such that the following distributions have statistical distance at most* $2^{-\Omega(\lambda)}$*:*

$$\left\{\mathsf{crs} \leftarrow \mathsf{CRSgen}(1^\lambda), (\mathsf{pk}, \mathsf{sk}, \mathsf{td}) \leftarrow \mathsf{Gen}(\mathsf{crs}), \mathbf{r} \leftarrow_{\mathsf{R}} \{0,1\}^{s(\lambda)} : \mathsf{CTsample}(\mathbf{r})\right\}.$$
$$\left\{\mathsf{crs} \leftarrow \mathsf{CRSgen}(1^\lambda), (\mathsf{pk}, \mathsf{sk}, \mathsf{td}) \leftarrow \mathsf{Gen}(\mathsf{crs}), \mathbf{m} \leftarrow_{\mathsf{R}} \mathbb{Z}_N^{\ell_2(\lambda)}, \mathsf{ct}^\star \leftarrow \mathsf{Enc}^\star_{\mathsf{pk}}(\mathbf{m}) : \mathsf{ct}^\star\right\}.$$

In Appendix A.1, we present the DJ LHE from the DCR assumption, and show that it is a hintable LHE. Now we present a variant of the packed Regev encryption scheme from the LWE assumption.

4.2 Packed-Regev, a Hintable LHE from LWE

We present a packed version of Regev encryption scheme [Reg05], and demonstrate that it can be used to satisfy our notion of a $(\ell_1, \ell_2, h, \alpha)$-hintable packed LHE, where ℓ_1 and ℓ_2 can be any polynomial and $h(\lambda)$ is a polynomial that that is larger than ℓ_1 but is independent of ℓ_2 That is, the hint is large in comparison to individual group elements, but we can pack in an arbitrary polynomial number of elements and still use the same hint size. We have $\alpha(\lambda) = \lambda + 1$.

As explained in the introduction, our construction is slightly different, but similar in spirit, to the Packed-Regev from [PVW08]. Just as in [PVW08], the idea is to individually encrypt the ℓ_2 different components of the message vector but reusing the *same* randomness $\mathbf{r}$ (but different parts

of the secret key) for the components. In contrast to [PVW08], as we do not want the length of the randomness to grow with ℓ_2, to prove security of the scheme, we add an extra smudging noise term $\mathbf{e}'$ to each encrypted component.

The hint for an encrypted message is a short pre-image of the ciphertext hear $\mathbf{Ar}$, where $\mathbf{r}$ is the randomness of the encryption. To enable efficiently recovering this hint, we will generate the lattice given in the public key together with a standard lattice trapdoor that enables sampling random short pre-images as in [Ajt96, GPV08, AP09, MP12].

To satisfy the properties of density and weak circuit privacy, we will rely on a extra noisy Packed-Regev encryption which proceeds just like the normal one but uses a much larger amount of randomness (so that it covers the whole set of strings for density, and so that it smudges the noises of evaluated ciphertexts for weak circuit privacy).

We will show how the Packed-Regev encryption scheme satisfies our notion of a hintable packed LHE.

Theorem 4.2. *Assume (subexponential) security of the LWE assumption holds. Then, for all polynomials ℓ_1, there exists some polynomial h such that for all polynomials ℓ_2, there exists a (subexponentially) secure $(\ell_1, \ell_2, h, \alpha)$-hintable packed LHE, with $\alpha(\lambda) = \lambda + 1$.*

4.2.1 Trapdoor Sampling

The Packed-Regev scheme makes use of the following lattice trapdoor mechanism: prior works [Ajt96, GPV08, AP09, MP12] show that there exist PPT algorithms $\mathsf{TrapGen}$ and $\mathsf{PreImSamp}$, an ensemble $\{\mathcal{D}_\lambda\}_{\lambda\in\mathbb{N}}$ of efficiently sampleable distributions over $\mathbb{Z}$ such that the following holds.

- $\underline{\mathsf{TrapGen}(1^\lambda, N, d)}$:

Given as input the security parameter $\lambda \in \mathbb{N}$, a modulus $N \in \mathbb{N}$, a dimension $d \in \mathbb{N}$, it outputs $(\mathbf{A}, T_\mathbf{A})$, where $\mathbf{A} \in \mathbb{Z}_N^{d\times m}$, $m \in \Theta(d\log(N))$ and $T_\mathbf{A}$ is a trapdoor. The matrix $\mathbf{A}$ is statistically close to uniform, that is, for all $\lambda, N, d \in \mathbb{N}$, the following distributions have statistical distance at most $2^{-\Omega(\lambda)}$: $\{(\mathbf{A}, T_\mathbf{A}) \leftarrow \mathsf{TrapGen}(1^\lambda, N, d) : \mathbf{A}\}$ and $\{\mathbf{A} \leftarrow_\mathsf{R} \mathbb{Z}_N^{\kappa\times m} : \mathbf{A}\}$.

- $\underline{\mathsf{PreImSamp}(\mathbf{A}, T_\mathbf{A}, \mathbf{t})}$:

Given as input the matrix $\mathbf{A}$, the trapdoor $T_\mathbf{A}$, a target vector $\mathbf{t} \in \mathbb{Z}_N^d$, it outputs $\mathbf{r} \in \mathbb{Z}_N^m$. For all $\lambda, d, N \in \mathbb{N}$, all $(\mathbf{A}, T_\mathbf{A})$ in the support of $\mathsf{TrapGen}(1^\lambda, N, d)$, all $\mathbf{t} \in \mathbb{Z}_N^d$, $\mathsf{PreImSamp}(\mathbf{A}, T_\mathbf{A}, \mathbf{t})$ outputs $\mathbf{r} \in \mathbb{Z}_N^m$ such that $\mathbf{Ar} = \mathbf{t} \in \mathbb{Z}_N^d$ and $\|\mathbf{r}\|_\infty < 2^{\lambda/2}$ with probability $1 - 2^{-\Omega(\lambda)}$ over its random coin.

We require the output $\mathbf{r}$ to follow some distribution that does not depend on the actual trapdoor $T_\mathbf{A}$, namely, for all $\lambda, d, N \in \mathbb{N}$, the following distributions have statistical distance at most $2^{-\Omega(\lambda)}$:

$$\{(\mathbf{A}, T_\mathbf{A}) \leftarrow \mathsf{TrapGen}(1^\lambda, N, d), \mathbf{r} \leftarrow_\mathsf{R} \mathcal{D}_\lambda^m, \mathbf{r}' \leftarrow_\mathsf{R} \mathsf{PreImSamp}(\mathbf{A}, T_\mathbf{A}, \mathbf{Ar}) : (\mathbf{r}', \mathbf{Ar})\}$$
$$\{(\mathbf{A}, T_\mathbf{A}) \leftarrow \mathsf{TrapGen}(1^\lambda, N, d), \mathbf{r} \leftarrow \mathcal{D}_\lambda^m : (\mathbf{r}, \mathbf{Ar})\}.$$

For our purposes, we want the distributions $\mathcal{D}_\lambda$ to be of smudging size. That is, for all polynomials $p(\cdot)$, all $\lambda, q \in \mathbb{N}$, the following distributions have statistical distance at most $2^{-\Omega(\lambda)}$:

$$\{r \leftarrow_\mathsf{R} \mathcal{D}_\lambda : r \in \mathbb{Z}_N\}$$
$$\{r \leftarrow_\mathsf{R} \mathcal{D}_\lambda : r + p(\lambda) \in \mathbb{Z}_N\}.$$

Finally, by the leftover hash lemma, since m is large enough and $\mathcal{D}_\lambda$ has enough entropy, for all $\lambda, N, q \in \mathbb{N}$, the following distributions have statistical distance at most $2^{-\Omega(\lambda)}$: $\{\mathbf{r} \leftarrow \mathcal{D}_\lambda^m, (\mathbf{A}, T_\mathbf{A}) \leftarrow \mathsf{TrapGen}(1^\lambda, N, d) : (\mathbf{A}, \mathbf{Ar})\}$ and $\{\mathbf{r} \leftarrow \mathcal{D}_\lambda^m, (\mathbf{A}, T_\mathbf{A}) \leftarrow \mathsf{TrapGen}(1^\lambda, N, d), \mathbf{u} \leftarrow_\mathsf{R} \mathbb{Z}_N^d : (\mathbf{A}, \mathbf{u})\}$.

4.2.2 The Construction

We now proceed to describing the Packed-Regev scheme, which is parameterized by polynomials ℓ_1 and ℓ_2. We denote the scheme by P-Regev$_{\ell_1,\ell_2}$.

• $\underline{\mathsf{CRSgen}(1^\lambda)}$:
It simply outputs $\mathsf{crs} = 1^\lambda$, i.e. there is no proper crs for that scheme.

• $\underline{\mathsf{Gen}(\mathsf{crs})}$:
Given as input $\mathsf{crs} = 1^\lambda$, it chooses $q = \{q_\kappa\}_{\kappa\in\mathbb{N}}$ with $q_\kappa = 2^{\kappa^c}$, a B_χ-bounded ensemble $\chi = \{\chi_\kappa\}_{\kappa\in\mathbb{N}}$ of efficiently sampleable distributions over $\mathbb{Z}$ with $B_\chi(\kappa) = \kappa$, and a polynomial $\kappa(\lambda) = \ell_1(\lambda)^{1/c}$, where $c \in (0, 1)$ is the constant from Definition 3.4, and $N = q_{\kappa(\lambda)}$. This choice of parameters ensures that the LWE assumption implies LWE holds for q, χ (i.e. the sequence q and the poynomial B_χ satisfy the requirement from Definition 3.4), and $N = 2^{\ell_1(\lambda)}$. We abuse notations and write $\kappa = \kappa(\lambda)$, $\chi = \chi_{\kappa(\lambda)}$ and $B_\chi = B_\chi(\kappa(\lambda))$ from here on.

Then, the algorithm samples $(\mathbf{A}, T_\mathbf{A}) \leftarrow \mathsf{TrapGen}(1^\lambda, N, \kappa)$, $\mathbf{S} \leftarrow \chi^{\ell_2\times\kappa}$, $\mathbf{E} \leftarrow \chi^{\ell_2\times m}$, and sets $\mathsf{pk} = (N, \mathbf{A}, \mathbf{SA} + \mathbf{E}) \in \mathbb{N} \times \mathbb{Z}_N^{\kappa\times m} \times \mathbb{Z}_N^{\ell_2\times m}$, $\mathsf{sk} = \mathbf{S}$ and $\mathsf{td} = T_\mathbf{A}$. It outputs $(\mathsf{pk}, \mathsf{sk}, \mathsf{td})$.

• $\underline{\mathsf{Enc}_\mathsf{pk}(\mathbf{x} \in \mathbb{Z}_N^\nu)}$:
Given the public pk, a vector $\mathbf{x} \in \mathbb{Z}_N^\nu$, it samples $\mathbf{R} \leftarrow [-1, 1]^{m\times\nu\ell_2}$, $\mathbf{E}' \leftarrow_\mathsf{R} [-2^{\lambda/2}, 2^{\lambda/2}]^{\ell_2\times\nu\ell_2}$ and outputs the ciphertext $\mathsf{ct} = \left(\mathbf{AR}, (\mathbf{SA} + \mathbf{E})\mathbf{R} + \mathbf{E}' + \mathbf{x}^\top \otimes \mathrm{Id}_{\ell_2}\right) \in \mathbb{Z}_N^{(\kappa+\ell_2)\times\nu\ell_2}$, where $\mathrm{Id}_{\ell_2} \in \mathbb{Z}_N^{\ell_2\times\ell_2}$ denotes the identity matrix, and $\mathbf{x}^\top \otimes \mathrm{Id}_{\ell_2} = \begin{pmatrix} \mathbf{x}^\top & 0 & \cdots \\ 0 & \mathbf{x}^\top & \\ \vdots & & \ddots \end{pmatrix} \in \mathbb{Z}_N^{\ell_2\times\nu\ell_2}$.

• $\underline{\mathsf{Enc}^\star_\mathsf{pk}(\mathbf{m} \in \mathbb{Z}_N^{\ell_2})}$:
Given the public pk, a message $\mathbf{m} \in \mathbb{Z}_N^{\ell_2}$, it samples $\mathbf{r}^\star \leftarrow_\mathsf{R} \mathcal{D}_\lambda^m$, where $\mathcal{D}_\lambda$ is the efficiently sampleable distribution over $\mathbb{Z}$ related to $\mathsf{TrapGen}$ and $\mathsf{PreImSamp}$; $\mathbf{e}^\star \leftarrow_\mathsf{R} [-2^\lambda, 2^\lambda]^{\ell_2}$, and outputs the noisy ciphertext $\mathsf{ct}^\star = \left(\mathbf{Ar}^\star, (\mathbf{SA} + \mathbf{E})\mathbf{r}^\star + \mathbf{e}^\star + \mathbf{m}\right) \in \mathbb{Z}_N^{\kappa+\ell_2}$.

• $\underline{\mathsf{Eval}(\mathsf{pk}, \mathsf{ct}, \mathsf{ct}^\star, \mathbf{y})}$:
Given the public pk, ciphertext $\mathsf{ct} \in \mathbb{Z}_N^{(\kappa+\ell_2)\times\nu\ell_2}$, noisy ciphertext $\mathsf{ct}^\star \in \mathbb{Z}_N^{\kappa+\ell_2}$ and a vector $\mathbf{y} \in [-1, 1]^{\nu\ell_2}$, it outputs the evaluated ciphertext $\mathsf{ct} \cdot \mathbf{y} + \mathsf{ct}^\star \in \mathbb{Z}_N^{\kappa+\ell_2}$.

• $\underline{\mathsf{SecHint}(\mathsf{td}, \mathsf{ct}^\star)}$:
Given as input the secret key sk and a (noisy or evaluated) ciphertext $\mathsf{ct}^\star \in \mathbb{Z}_N^{\kappa+\ell_2}$ of the form $(\mathbf{t}, \mathbf{z})$ where $\mathbf{t} \in \mathbb{Z}_N^\kappa$ and $\mathbf{z} \in \mathbb{Z}_N^{\ell_2}$, it samples $\rho \leftarrow \mathsf{PreImSamp}(\mathbf{A}, T_\mathbf{A}, \mathbf{t})$ and outputs the hint $\rho \in \mathbb{Z}_N^m$.

• $\underline{\mathsf{PubHint}(\mathsf{pk}, r)}$:
Given as input the public key pk and the random coins $r = (\mathbf{r}^\star, \mathbf{e}^\star)$ where $\mathbf{r}^\star \in \mathcal{D}_\lambda^m$, $\mathbf{e}^\star \in [-2^\lambda, 2^\lambda]^{\ell_2}$ used to produce a noisy ciphertext, it outputs $\mathbf{r}^\star \in \mathbb{Z}_N^m$.

• $\underline{\mathsf{Rec}(\mathsf{pk}, \mathsf{ct}^\star, \rho)}$:
Given as input the public key pk, a (noisy or evaluated) ciphertext $\mathsf{ct}^\star \in \mathbb{Z}_N^{\kappa+\ell_2}$ of the form $\mathsf{ct} = (\mathbf{t}, \mathbf{z})$ where $\mathbf{t} \in \mathbb{Z}_N^\kappa$, $\mathbf{z} \in \mathbb{Z}_N^{\ell_2}$ and a hint $\rho \in \mathbb{Z}_N^m$, it outputs $\mathbf{d} = \mathbf{z} - (\mathbf{SA} + \mathbf{E})\rho \in \mathbb{Z}_N^{\ell_2}$.

• $\underline{\mathsf{Dec}_{\mathsf{sk}}(\mathsf{ct}^\star)}$:
Given as input the secret key sk and a (noisy or evaluated) ciphertext $\mathsf{ct}^\star = (\mathbf{t}, \mathbf{z})$ with $\mathbf{t} \in \mathbb{Z}_N^\kappa$, $\mathbf{z} \in \mathbb{Z}_N^{\ell_2}$, it outputs $\mathbf{d} = \mathbf{z} - \mathbf{St} \in \mathbb{Z}_N^{\ell_2}$.

The proof of Theorem 4.2 follows from the propositions and theorem below (which demonstrate that Packed-Regev Scheme satisfies the desired properties of a hintable packed LHE, as well as security).

Proposition 6 (λ+1-approximate correctness)**.** *The LHE presented above satisfies λ+1-approximate correctness, as defined in Property 4.1.*

Proof: A ciphertext $\mathsf{ct}^\star$ in the support of $\mathsf{Enc}^\star_{\mathsf{pk}}(\mathbf{m})$ is of the form $\mathsf{ct}^\star = (\mathbf{t}, \mathbf{z})$ with $\mathbf{t} = \mathbf{Ar}^\star \in Z_q^\kappa$ and $\mathbf{z} = (\mathbf{SA} + \mathbf{E})\mathbf{r}^\star + \mathbf{e}^\star + \mathbf{m} \in \mathbb{Z}_N^{\ell_2}$. The vector $\mathbf{d} \in \mathbb{Z}_N^{\ell_2}$ output by the decryption is of the form $\mathbf{m}+\mathsf{noise}$ where $\mathsf{noise} = \mathbf{Er}^\star+\mathbf{e}^\star$. For all $\mathbf{E} \in \mathbb{Z}_N^{\ell_2 \times m}$ such that $\|\mathbf{E}\|_\infty \leq B_\chi$, with probability $1-2^{-\Omega(\lambda)}$ over the choices of $\mathbf{e}^\star \leftarrow [-2^\lambda, 2^\lambda]^{\ell_2}$ and $\mathbf{r}^\star \leftarrow \mathcal{D}_\lambda^m$, we have $\|\mathsf{noise}\|_\infty \leq 2^\lambda + B_\chi m 2^{\lambda/2} \leq 2^{\lambda+1}$. □

Proposition 7 (Linear Homomorphism)**.** *The LHE presented above satisfies Property 4.2.*

Proof: The ciphertext produced by $\mathsf{Enc}_{\mathsf{pk}}(\mathbf{x})$ has the form $\mathsf{ct} = \left(\mathbf{AR}, (\mathbf{SA} + \mathbf{E})\,\mathbf{R} + \mathbf{E}' + \mathbf{x}^\top \otimes \mathrm{Id}_{\ell_2}\right) \in \mathbb{Z}_N^{(\kappa+\ell_2)\times\nu\ell_2}$, and $\mathsf{ct}^\star = (\mathbf{Ar}^\star, (\mathbf{SA} + \mathbf{E})\,\mathbf{r}^\star + \mathbf{e}^\star + \mathbf{x}^\star)$. For all $\mathbf{y} \in [-1, 1]^{\nu\ell_2}$, $\mathsf{Eval}(\mathsf{pk}, \mathsf{ct}, \mathsf{ct}^\star, \mathbf{y})$ outputs the evaluated ciphertext $\mathsf{ct}_{\mathbf{y}} = \Big(\mathbf{A}(\mathbf{Ry}+\mathbf{r}^\star), (\mathbf{SA}+\mathbf{E})(\mathbf{Ry}+\mathbf{r}^\star)+\mathbf{E}'\mathbf{y}+\mathbf{e}^\star+\mu\Big) \in \mathbb{Z}_N^{\kappa+\ell_2}$ where $\mu = (m_1 + \mathbf{x}^\top\mathbf{y}_1, \ldots, m_{\ell_2} + \mathbf{x}^\top\mathbf{y}_{\ell_2}) \in \mathbb{Z}_N^{\ell_2}$ which is in the support of $\mathsf{Enc}^\star_{\mathsf{pk}}(\mu)$. □

Proposition 8 (λ+1-approximate correctness of the secret hints)**.** *The LHE presented above satisfies $\lambda + 1$-approximate correctness of the secret hints, as defined in Property 4.3.*

Proof: For all messages $\mathbf{m} \in \mathbb{Z}_N^{\ell_2}$, $\mathsf{Enc}^\star\mathsf{pk}(\mathbf{m})$ is of the form $\mathsf{ct}^\star = (\mathbf{t}, \mathbf{z}) \in \mathbb{Z}_N^{\kappa\times\ell_2}$, where $\mathbf{t} = \mathbf{Ar}^\star$ and $\mathbf{z} = (\mathbf{SA} + \mathbf{E})\,\mathbf{r}^\star + \mathbf{e}^\star + \mathbf{x}^\star\big)$. The algorithm $\mathsf{SecHint}$ computes $\rho \leftarrow \mathsf{PreImSamp}(\mathbf{A}, T_{\mathbf{A}}, \mathbf{t})$, which is such that $\mathbf{A}\rho = \mathbf{t}$. Next, the algorithm Rec computes $\mathbf{d} = \mathbf{z} - (\mathbf{SA} + \mathbf{E})\,\rho = \mathbf{m} + \mathsf{noise}$, where $\mathsf{noise} = \mathbf{e}^\star + \mathbf{E}(\mathbf{r}^\star - \rho)$. For all $\mathbf{E} \in \mathbb{Z}_N^{\ell_2\times m}$ such that $\|\mathbf{E}\|_\infty \leq B_\chi$, with probability $1 - 2^{-\Omega(\lambda)}$ over the choices of $\mathbf{e}^\star \leftarrow [-2^\lambda, 2^\lambda]^{\ell_2}$, $\mathbf{r}^\star \leftarrow \mathcal{D}_\lambda^m$ and the random coins used to produce ρ, we have $\|\mathsf{noise}\|_\infty \leq 2^\lambda + 2mB_\chi 2^{\lambda/2} \leq 2^{\lambda+1}$. □

Proposition 9 (Equivalence between public and secret hints)**.** *The LHE presented above satisfies Property 4.4.*

Proof: We aim at proving that for all $\lambda \in \mathbb{N}$, all messages $\mathbf{m} \in \mathbb{Z}_N^{\ell_2}$, the following distributions have statistical distance at most $2^{-\Omega(\lambda)}$:

$$\mathcal{D}_0 = \left\{ \begin{array}{l} \big(\mathsf{pk} = (\mathbf{A}, \mathbf{SA} + \mathbf{E}), \mathsf{sk} = \mathbf{S}, \mathsf{td} = T_{\mathbf{A}}\big) \leftarrow \mathsf{Gen}(1^\lambda), \mathbf{r}^\star \leftarrow \mathcal{D}_\lambda^m, \mathbf{e}^\star \leftarrow_{\mathsf{R}} [-2^\lambda, 2^\lambda]^{\ell_2} \\ \rho \leftarrow \mathsf{PreImSamp}(\mathbf{A}, T_{\mathbf{A}}, \mathbf{Ar}^\star) : (\mathbf{Ar}^\star, (\mathbf{SA} + \mathbf{E})\mathbf{r}^\star + \mathbf{e}^\star + \mathbf{m}, \rho) \end{array} \right\}$$

$$\mathcal{D}_1 = \left\{ \begin{array}{l} \big(\mathsf{pk} = (\mathbf{A}, \mathbf{SA} + \mathbf{E}), \mathsf{sk} = \mathbf{S}, \mathsf{td} = T_{\mathbf{A}}\big) \leftarrow \mathsf{Gen}(1^\lambda), \mathbf{r}^\star \leftarrow \mathcal{D}_\lambda^m \\ \mathbf{e}^\star \leftarrow_{\mathsf{R}} [-2^\lambda, 2^\lambda]^{\ell_2} : (\mathbf{Ar}^\star, (\mathbf{SA} + \mathbf{E})\mathbf{r}^\star + \mathbf{e}^\star + \mathbf{m}, \mathbf{r}^\star) \end{array} \right\}$$

By Lemma 2.2 (smudging), distribution $\mathcal{D}_0$ has statistical distance at most $2^{-\Omega(\lambda)}$ with $\mathcal{D}_0' = \{(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda), \mathbf{r}^\star \leftarrow \mathcal{D}_\lambda^m, \mathbf{e}^\star \leftarrow_{\mathsf{R}} \big[-2^\lambda, 2^\lambda\big]^{\ell_2}, \rho \leftarrow \mathsf{PreImSamp}(\mathbf{A}, T_{\mathbf{A}}, \mathbf{Ar}^\star) : \big(\mathbf{Ar}^\star, \mathbf{SAr}^\star +$

$\mathbf{e}^\star + \mathbf{m}, \rho)\}$. By the property of $\mathsf{PreImSamp}$, $\mathcal{D}'_0$ has statistical distance at most $2^{-\Omega(\lambda)}$ with $\mathcal{D}'_1 = \{(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda), \mathbf{r}^\star \leftarrow \mathcal{D}_\lambda^m, \mathbf{e}^\star \leftarrow_\mathsf{R} \left[-2^\lambda, 2^\lambda\right]^{\ell_2} : \left(\mathbf{A}\mathbf{r}^\star, \mathbf{S}\mathbf{A}\mathbf{r}^\star + \mathbf{e}^\star + \mathbf{m}, \mathbf{r}^\star\right)\}$. Finally, using smudging again, we have $\mathcal{D}'_1 \approx_s \mathcal{D}_1$. □

Proposition 10 (Succinctness of hints)**.** *The LHE presented above satisfies $h(\lambda) = (\lambda/2 + 1)m$ succinctness, where $m = 2\kappa \log(q) + 2\lambda$.*

Proof: With probability $1 - 2^{-\Omega(\lambda)}$ over the random coins of $\mathsf{SecHint}$, the hint output by $\mathsf{SecHint}$ belong to $[-2^{\lambda/2}, 2^{\lambda/2}]^m$. □

Proposition 11 (Density of the noisy ciphertexts)**.** *The LHE presented above satisfies Property 4.7.*

Proof: For all $\mathbf{m} \in \mathbb{Z}_N^{\ell_2}$, a noisy ciphertext $\mathsf{ct}^\star \leftarrow_\mathsf{R} \mathsf{Enc}^\star_\mathsf{pk}(\mathbf{m})$ is of the form $\mathsf{ct}^\star = (\mathbf{A}\mathbf{r}^\star, (\mathbf{S}\mathbf{A}+\mathbf{E})\mathbf{r}^\star + \mathbf{e}^\star + \mathbf{m}) \in \mathbb{Z}_N^{\kappa+\ell_2}$. When $\mathbf{m} \leftarrow_\mathsf{R} \mathbb{Z}_N^{\ell_2}$, the second part of the ciphertext is uniformly random over $\mathbb{Z}_N^{\ell_2}$. That is, for all $\lambda \in \mathbb{N}$, for all pairs $(\mathsf{pk}, \mathsf{sk})$ in the support of $\mathsf{Gen}(1^\lambda)$, the following distributions are identical: $\{\mathbf{m} \leftarrow_\mathsf{R} \mathbb{Z}_N^{\ell_2}, \mathsf{ct}^\star \leftarrow \mathsf{Enc}^\star_\mathsf{pk}(\mathbf{m}) : \mathsf{ct}^\star\}$ and $\{\mathbf{r}^\star \leftarrow_\mathsf{R} \mathcal{D}_\lambda^m, \mathbf{w} \leftarrow_\mathsf{R} \mathbb{Z}_N^{\ell_2} : (\mathbf{A}\mathbf{r}^\star, \mathbf{w})\}$. We conclude the proof using the properties of $\mathsf{PreImSamp}$, which imply that the following distributions have statistical distance at most $2^{-\Omega(\lambda)}$: $\{\mathbf{A} \leftarrow_\mathsf{R} \mathbb{Z}_N^{\kappa\times m}, \mathbf{r}^\star \leftarrow_\mathsf{R} \mathcal{D}_\lambda^m : (\mathbf{A}, \mathbf{A}\mathbf{r}^\star)\}$ and $\{\mathbf{A} \leftarrow_\mathsf{R} \mathbb{Z}_N^{\kappa\times m}, \mathbf{u} \leftarrow_\mathsf{R} \mathbb{Z}_N^{\kappa} : (\mathbf{A}, \mathbf{u})\}$. □

Proposition 12 (Weak circuit privacy)**.** *The LHE presented above satisfies Property 4.6.*

Proof: For all $\mathbf{x} \in \mathbb{Z}_N^\nu$, $\mathbf{y} \in [-1, 1]^{\nu\ell_2}$, we aim at proving the following distributions have statistical distance at most $2^{-\Omega(\lambda)}$:

$$\mathcal{D}_0 = \left\{ \ (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda), \mathsf{ct} \leftarrow \mathsf{Enc}_\mathsf{pk}(\mathbf{x}), \mathsf{ct}^\star \leftarrow \mathsf{Enc}^\star_\mathsf{pk}(\mathbf{m}), \mathsf{ct}_\mathbf{y} = \mathsf{Eval}(\mathsf{pk}, \mathsf{ct}, \mathsf{ct}^\star, \mathbf{y}) : (\mathsf{ct}, \mathsf{ct}^\star, \mathsf{ct}_\mathbf{y}) \ \right\}$$

$$\mathcal{D}_1 = \left\{ \begin{array}{l} (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda), \mathsf{ct} \leftarrow \mathsf{Enc}_\mathsf{pk}(\mathbf{x}) \\ \mathsf{ct}_\mathbf{y} \leftarrow \mathsf{Enc}^\star_\mathsf{pk}(\mu), \mathsf{ct}^\star = \mathsf{Eval}(\mathsf{pk}, \mathsf{ct}, \mathsf{ct}_\mathbf{y}, -\mathbf{y}) : (\mathsf{ct}, \mathsf{ct}^\star, \mathsf{ct}_\mathbf{y}) \end{array} \right\},$$

where $\mu = (m_1 + \mathbf{x}^\top \mathbf{y}_1, \ldots, m_{\ell_2} + \mathbf{x}^\top \mathbf{y}_{\ell_2}) \in \mathbb{Z}_N^{\ell_2}$.

In distribution $\mathcal{D}_0$, we have:

- $\mathsf{ct} = \left(\mathbf{A}\mathbf{R}, (\mathbf{S}\mathbf{A} + \mathbf{E})\,\mathbf{R} + \mathbf{E}' + \mathbf{x}^\top \otimes \mathrm{Id}_{\ell_2}\right)$,
- $\mathsf{ct}^\star = (\mathbf{A}\mathbf{r}^\star, (\mathbf{S}\mathbf{A} + \mathbf{E})\,\mathbf{r}^\star + \mathbf{e}^\star + \mathbf{x}^\star)$,
- $\mathsf{ct}_\mathbf{y} = (\mathbf{A}(\mathbf{R}\mathbf{y} + \mathbf{r}^\star), (\mathbf{S}\mathbf{A} + \mathbf{E})\,(\mathbf{R}\mathbf{y} + \mathbf{r}^\star) + \mathbf{E}'\mathbf{y} + \mathbf{e}^\star + \mu)$.

We show that it has statistical distance $2^{-\Omega(\lambda)}$ from $\mathcal{D}_1$ by a series of claims.

Claim 1. For all $\mathbf{y} \in [-1, 1]^{\nu\ell_2}$, the following distributions have statistical distance $2^{-\Omega(\lambda)}$:

$$\left\{\mathbf{r}^\star \leftarrow_\mathsf{R} \mathcal{D}_\lambda^m, \mathbf{R} \leftarrow_\mathsf{R} [-1, 1]^{m\times\nu\ell_2} : (\mathbf{R}, \mathbf{r}^\star, \mathbf{r}^\star + \mathbf{R}\mathbf{y})\right\}$$
$$\left\{\mathbf{r}^\star \leftarrow_\mathsf{R} \mathcal{D}_\lambda^m, \mathbf{R} \leftarrow_\mathsf{R} [-1, 1]^{m\times\nu\ell_2} : (\mathbf{R}, \mathbf{r}^\star - \mathbf{R}\mathbf{y}, \mathbf{r}^\star)\right\},$$

by the properties of $\mathsf{PreImSamp}$.

Claim 2. For all $\mathbf{y} \in [-1, 1]^{\nu\ell_2}$, the following distributions have statistical distance $2^{-\Omega(\lambda)}$:

$$\left\{\mathbf{e}^\star \leftarrow_\mathsf{R} \left[-2^\lambda, 2^\lambda\right]^{\ell_2}, \mathbf{E}' \leftarrow [-2^{\lambda/2}, 2^{\lambda/2}]^{\ell_2\times\nu\ell_2} : (\mathbf{E}', \mathbf{e}^\star, \mathbf{e}^\star + \mathbf{E}'\mathbf{y})\right\}$$
$$\left\{\mathbf{e}^\star \leftarrow_\mathsf{R} \left[-2^\lambda, 2^\lambda\right]^{\ell_2}, \mathbf{E}' \leftarrow [-2^{\lambda/2}, 2^{\lambda/2}]^{\ell_2\times\nu\ell_2} : (\mathbf{E}', \mathbf{e}^\star - \mathbf{E}'\mathbf{y}, \mathbf{e}^\star)\right\}.$$

by Lemma 2.2 (smudging).

Claim 1 and Claim 2 imply the following claim.

Claim 3. For all $\mathbf{y} \in [-1,1]^{\nu\ell_2}$, the following distributions have statistical distance $2^{-\Omega(\lambda)}$:

$$\mathcal{D}'_0 = \left\{ \begin{array}{l} \mathbf{e}^\star \leftarrow_{\mathsf{R}} \left[-2^\lambda, 2^\lambda\right]^{\ell_2}, \mathbf{E}' \leftarrow [-2^{\lambda/2}, 2^{\lambda/2}]^{\ell_2 \times \nu\ell_2}, \mathbf{r}^\star \leftarrow_{\mathsf{R}} \mathcal{D}_\lambda^m \\ \mathbf{R} \leftarrow_{\mathsf{R}} [-1,1]^{m \times \nu\ell_2} : \left(\mathbf{R}, \mathbf{r}^\star, \mathbf{r}^\star + \mathbf{R}\mathbf{y}, \mathbf{E}', \mathbf{e}^\star, \mathbf{e}^\star + \mathbf{E}'\mathbf{y}\right) \end{array} \right\}$$

$$\mathcal{D}'_1 = \left\{ \begin{array}{l} \mathbf{e}^\star \leftarrow_{\mathsf{R}} \left[-2^\lambda, 2^\lambda\right]^{\ell_2}, \mathbf{E}' \leftarrow [-2^{\lambda/2}, 2^{\lambda/2}]^{\ell_2 \times \nu\ell_2}, \mathbf{r}^\star \leftarrow_{\mathsf{R}} \mathcal{D}_\lambda^m \\ \mathbf{R} \leftarrow_{\mathsf{R}} [-1,1]^{m \times \nu\ell_2} : \left(\mathbf{R}, \mathbf{r}^\star - \mathbf{R}\mathbf{y}, \mathbf{r}^\star, \mathbf{E}', \mathbf{e}^\star - \mathbf{E}'\mathbf{y}, \mathbf{e}^\star\right) \end{array} \right\}.$$

Now we prove the following claim, that will conclude the proof of Proposition 12.

Claim 4. There exists a (possibly inefficient) simulator $\mathcal{S}$ that given as input $\mathbf{v} \in \left(\mathbb{Z}_N^{m \times (\nu\ell_2+2)} \times \mathbb{Z}_N^{\ell_2 \times (\nu\ell_2+2)}\right.$, outputs a tuple $(\mathsf{ct}, \mathsf{ct}^\star, \mathsf{ct}_\mathbf{y}) \in \mathbb{Z}_N^{(\kappa+\ell_2)\times\nu\ell_2} \times \left(\mathbb{Z}_N^{\kappa+\ell_2}\right)^2$, such that when $\mathcal{S}$ is fed with an input from distribution $\mathcal{D}'_0$, it produces an output following the distribution $\mathcal{D}_0$, whereas when fed with an input coming from distribution $\mathcal{D}'_1$, it produces an output following the distribution $\mathcal{D}_1$.

Given as input $\left(\mathbf{R}, \mathbf{r}_1, \mathbf{r}_2, \mathbf{E}', \mathbf{e}_1, \mathbf{e}_2\right)$, $\mathcal{S}$ computes:

- $\mathsf{ct} = \left(\mathbf{AR}, (\mathbf{SA}+\mathbf{E})\,\mathbf{R} + \mathbf{E}' + \mathbf{x}^\top \otimes \mathrm{Id}_{\ell_2}\right)$,
- $\mathsf{ct}^\star = (\mathbf{A}\mathbf{r}_1, (\mathbf{SA}+\mathbf{E})\,\mathbf{r}_1 + \mathbf{e}_1 + \mathbf{m})$,
- $\mathsf{ct}_\mathbf{y} = (\mathbf{A}\mathbf{r}_2, (\mathbf{SA}+\mathbf{E})\,\mathbf{r}_2 + \mathbf{e}_2 + \mu)$.

□

Theorem 4.3 (Security)**.** *Assume (subexponential) security of the LWE assumption holds. Then, for all polynomials ℓ_1 and ℓ_2,* P-Regev$_{\ell_1,\ell_2}$ *is (subexponentially) secure.*

Proof: In a nutshell, Regev encryption uses the LWE assumption to switch the LWE samples from the public key $\mathbf{SA}+\mathbf{E}$ to uniformly random, then use the leftover hash lemma to extract the entropy from the randomness used to encrypt the messages. The problem here is that for succinctness, we use small randomness, and large messages: the randomness does not hold enough entropy to hide the messages. Instead, we use the randomness and extra smudging noise to generate fresh LWE samples, and then hide the messages. We now provide the formal proof, which uses the hybrids experiment listed below.

• $\underline{\mathcal{H}^0_\lambda}$: as per Definition 2.8. Namely, the experiment generates $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$ where $\mathsf{pk} = (N, \mathbf{A}, \mathbf{SA}+\mathbf{E})$, sends pk to the adversary who chooses a pair $(\mathbf{x}^0, \mathbf{x}^1) \in \mathbb{Z}_N^{\nu_0} \times \mathbb{Z}_N^{\nu_1}$ (wlog. we can assume $\nu_0 = \nu_1 = \nu$). The experiment samples $b \leftarrow_{\mathsf{R}} \{0,1\}$ and computes the challenge ciphertext as $\mathsf{ct} = (\mathbf{AR}, (\mathbf{SA}+\mathbf{E})\mathbf{R} + \mathbf{E}' + \mathbf{x}^b \otimes \mathrm{Id}_{\ell_2})$, which is sent to the adversary, who wins if it guesses b successfully.

• $\underline{\mathcal{H}^1_\lambda}$: this experiment is the same as $\mathcal{H}^0_\lambda$ except the challenge ciphertext is now of the form $\mathsf{ct} = (\mathbf{AR}, \mathbf{SAR} + \mathbf{E}' + \mathbf{x}^b \otimes \mathrm{Id}_{\ell_2})$. Recall that $\mathbf{E} \leftarrow_{\mathsf{R}} \chi^{\ell_2 \times m}$ where χ is B_χ-bounded, for a polynomial B_χ, $\mathbf{R} \leftarrow_{\mathsf{R}} [-1,1]^{m \times \ell_2}$ for a polynomial m and $\mathbf{E}' \leftarrow_{\mathsf{R}} [-2^{\lambda/2}, 2^{\lambda/2}]^{\ell_2 \times \ell_2}$. Thus, we can use Lemma 2.2 (smudging) to argue that $\mathbf{E}' + \mathbf{ER} \approx_s \mathbf{E}'$ with statistical distance $2^{-\Omega(\lambda)}$. The first distribution corresponds to $\mathcal{H}^0_\lambda$ (with pre and post-processing), whereas the second distribution corresponds to $\mathcal{H}^1_\lambda$ (with pre and post-processing).

• $\underline{\mathcal{H}^2_\lambda}$: this experiment is the same as $\mathcal{H}^1_\lambda$, except the challenge ciphertext is now of the form $\mathsf{ct} = (\mathbf{U}, \mathbf{SU} + \mathbf{E}' + \mathbf{x}^b \otimes \mathrm{Id}_{\ell_2})$, where $\mathbf{U} \leftarrow_{\mathsf{R}} \mathbb{Z}_N^{\kappa \times \ell_2}$. The fact that $\{\mathcal{H}^1_\lambda\}_{\lambda\in\mathbb{N}} \approx_s \{\mathcal{H}^2_\lambda\}_{\lambda\in\mathbb{N}}$ follows

readily from the leftover hash lemma, which states that the following distributions have statistical distance $2^{-\Omega(\lambda)}$:

$$\{\mathbf{A} \leftarrow_{\mathsf{R}} \mathbb{Z}_N^{\kappa \times m}, \mathbf{R} \leftarrow_{\mathsf{R}} [-1,1]^{m \times \ell_2} : (\mathbf{A}, \mathbf{AR})\}$$
$$\{\mathbf{A} \leftarrow_{\mathsf{R}} \mathbb{Z}_N^{\kappa \times m}, \mathbf{U} \leftarrow_{\mathsf{R}} \mathbb{Z}_N^{\kappa \times \ell_2} : (\mathbf{A}, \mathbf{U})\}.$$

The first distribution corresponds to $\mathcal{H}_\lambda^1$ (with post-processing), whereas the second distribution corresponds to $\mathcal{H}_\lambda^2$ (with the same post-processing).

• $\underline{\mathcal{H}_\lambda^3}$: this experiment is the same as $\mathcal{H}_\lambda^2$, except the challenge ciphertext is now of the form $\mathsf{ct} = (\mathbf{U}, \mathbf{W})$, where $\mathbf{W} \leftarrow_{\mathsf{R}} \mathbb{Z}_N^{\kappa \times \nu \ell_2}$, and the public key is now of the form $\mathsf{pk} = (N, \mathbf{A}, \mathbf{V})$ where $\mathbf{V} \leftarrow_{\mathsf{R}} \mathbb{Z}_N^{\ell_2 \times m}$. We show that $\{\mathcal{H}_\lambda^2\}_{\lambda \in \mathbb{N}} \approx_c \{\mathcal{H}_\lambda^3\}_{\lambda \in \mathbb{N}}$. This holds by the properties of $\mathsf{TrapGen}$, which state that the matrix $\mathbf{A}$ is statistically close to uniform over $\mathbb{Z}_N^{\kappa \times m}$. Then, we rely on the LWE assumption, which implies that $(\mathbf{A}, \mathbf{SA} + \mathbf{E}, \mathbf{U}, \mathbf{SU} + \mathbf{E}' + \mathbf{x}^b \otimes \mathrm{Id}_{\ell_2}) \approx_c (\mathbf{A}, \mathbf{V}, \mathbf{U}, \mathbf{W} + \mathbf{x}^b \otimes \mathrm{Id}_{\ell_2}) \equiv (\mathbf{A}, \mathbf{V}, \mathbf{U}, \mathbf{W})$, where $\mathbf{W} \leftarrow_{\mathsf{R}} \mathbb{Z}_N^{\kappa \times \nu \ell_2}$ and $\mathbf{V} \leftarrow_{\mathsf{R}} \mathbb{Z}_N^{\ell_2 \times m}$. We conclude by noting that in the experiment $\mathcal{H}_\lambda^3$, the adversary's view does not depend on the random bit b. □

5 Constructing XiO for $\mathsf{P}^{\log}/\mathsf{poly}$

We present a modular construction of XiO $\mathsf{P}^{\log}/\mathsf{poly}$ from the GSW FHE scheme for circuits of depth δ, denoted by GSW_δ, for sufficiently sufficiently large δ, and any $(\ell_1, \ell_2, h, \alpha)$-hintable packed LHE for sufficiently small h, α and sufficiently large ℓ_1 and ℓ_2 —recall that h measures the LHE succinctness, α quantifies the approximate correctness, while ℓ_1, ℓ_2 measure the plaintext size, or "batching capacity" of the scheme; ℓ_1 intuitively represents how many bits can be packed in a scalar, i.e. how large is the modulus in use, whereas ℓ_2 measures how many such scalars are recovered when decrypting one ciphertext. As a warm up to our main theorem, we first prove the IND-security of our XiO construction from 2-circular SRL security of the GSW scheme and the hintable LHE, where 2-circular security is defined similarly as 1-circular security. Later, we improve this result by showing that 1-circular SRL security of the GSW scheme suffices.

Outline. In the rest of this section, we abstract out 2 additional properties of the GSW FHE that we will rely on to enable a modular proof. This is done in Section 5.1. Then, we present our XiO construction in Section 5.2. Afterwards, in Section 5.3 we define the notion of 2-circular SRL security and prove our first main theorem, namely, the IND-security of our XiO assuming the 2-circular SRL security of the GSW scheme and the hintable LHE. Then, in Section 5.4, we show that 1-circular SRL security of GSW is sufficient to prove the security of our XiO. Finally, we state our main theorem in Section 5.5.

5.1 Additional properties for GSW

Weak Circuit Privacy of GSW_δ. As mentionned in the introduction, we will rely on the fact that the GSW encryption scheme also satisfies a notion of "weak circuit privacy" similar to the one defined for LHE. More precisely, we show that GSW satisfies a property that involves a public-key algorithm that re-randomizes evaluated ciphertext so that they look like fresh ciphertexts from the support of the noise encryption algorithm $\mathsf{Enc}^\star$. Namely, we show that there exists a PPT algorithm ReRand that takes as input the public key pk, an evaluated ciphertext ct, some random coins $\mathbf{r}^\star \in \mathcal{R}^\star$, and outputs an evaluated ciphertext ct computed as described below.

• $\underline{\mathsf{ReRand}(\mathsf{pk}, \mathsf{ct}; \mathbf{r}^\star)}$:
Given $\mathsf{pk} = (B, \mathbf{U}, \mathbf{G})$, $\mathsf{ct} \in \{0,1\}^w$ and $\mathbf{r}^\star \leftarrow_{\mathsf{R}} [-B^\star, B^\star]^m$, it computes $\mathsf{ct}' \in \mathbb{Z}_N^{\kappa+1}$ whose binary decomposition is ct, computes $\widetilde{\mathsf{ct}} = \mathsf{ct}' + \mathbf{U}\mathbf{r}^\star \in \mathbb{Z}_N^{\kappa+1}$, and outputs the re-randomized ciphertext $\mathsf{BD}(\widetilde{\mathsf{ct}}) \in \{0,1\}^w$.

Theorem 5.1 (weak circuit privacy)**.** *For all polynomials ν, ℓ, δ, all $\lambda \in \mathbb{N}$, all crs containing a modulus $N \in \mathbb{N}$ such that $N \geq 2^{2\lambda}B$, where B is an upper-bound on the noise obtained from homomorphically evaluating circuits of depth at most $\delta(\lambda)$, all pairs $(\mathsf{pk}, \mathsf{sk})$ in the support of $\mathsf{Gen}(\mathsf{crs})$, all messages $\mu \in \{0,1\}^{\nu(\lambda)}$, all depth-$\delta(\lambda)$ circuits $f_1, \ldots, f_{\ell(\lambda)} : \{0,1\}^{\nu(\lambda)} \to \mathbb{Z}_N$, the following distributions have statistical distance at most $2^{-\Omega(\lambda)}$:*

$$\mathcal{D}_0 : \left\{ \begin{array}{l} \mathsf{ct} \leftarrow \mathsf{Enc}_{\mathsf{pk}}(\mu), \forall j \in [\ell(\lambda)], \mathsf{ct}_{f_j} = \mathsf{Eval}'(\mathsf{pk}, f_j, 0, \mathsf{ct}), \mathbf{r}_j^\star \leftarrow_{\mathsf{R}} \mathcal{R}^\star \\ \mathsf{ct}_{f_j}^\star = \mathsf{ReRand}(\mathsf{pk}, \mathsf{ct}_{f_j}; \mathbf{r}^\star) : \left(\mathsf{ct}, (\mathbf{r}_j^\star, \mathsf{ct}_{f_j}^\star)_{j \in [\ell(\lambda)]}\right) \end{array} \right\}$$

$$\mathcal{D}_1 : \left\{ \begin{array}{l} r \leftarrow_{\mathsf{R}} \mathcal{R}^{\nu(\lambda)}, \mathsf{ct} = \mathsf{Enc}_{\mathsf{pk}}(\mu; r), \forall j \in [\ell(\lambda)], \mathbf{r}_j^\star \leftarrow_{\mathsf{R}} \mathcal{R}^\star, \mathsf{ct}_{f_j}^\star = \mathsf{Enc}_{\mathsf{pk}}^\star(f_j(\mu); \mathbf{r}_j^\star) \\ \mathbf{r}_{f_j} = \mathsf{Eval}_{\mathsf{rand}}(\mathsf{pk}, f_j, r, \mu) : \left(\mathsf{ct}, (\mathbf{r}_j^\star - \mathbf{r}_{f_j}, \mathsf{ct}_{f_j}^\star)_{j \in [\ell(\lambda)]}\right) \end{array} \right\}$$

Proof: By batch correctness of the scheme, for all $j \in [\ell(\lambda)]$, the evaluated ciphertext is of the form $\mathsf{ct}_{f_j} = \left(\mathbf{A}\mathbf{r}_{f_j}, (\mathbf{s}^\top\mathbf{A} + \mathbf{e}^\top)\mathbf{r}_{f_j} + f_j(\mu)\right) \in \mathbb{Z}_N^{\kappa+1}$, and the re-randomized ciphertext is of the form $\mathsf{ct}_{f_j}^\star = \left(\mathbf{A}(\mathbf{r}_{f_j} + \mathbf{r}_j^\star), (\mathbf{s}^\top\mathbf{A} + \mathbf{e}^\top)(\mathbf{r}_{f_j} + \mathbf{r}_j^\star) + f_j(\mu)\right) \in \mathbb{Z}_N^{\kappa+1}$, where $\|\mathbf{r}_{f_j}\|_\infty < (w+1)^\delta \lceil \log(N) \rceil = 2^{-\lambda}B^\star$ and $\mathbf{r}_j^\star \leftarrow_{\mathsf{R}} [-B^\star, B^\star]^m$. By Lemma 2.2 (smudging), the following distributions have statistical distance at most $2^{-\lambda}$:

$$\left(\mathbf{r}_j^\star\right)_{j \in [\ell]} \approx_s \left(\mathbf{r}_j^\star - \mathbf{r}_{f_j}\right)_{j \in [\ell]}.$$

The leftmost distribution corresponds to $\mathcal{D}_0$ (with pre and post-processing), whereas the rightmost distribution corresponds to $\mathcal{D}_1$ (with pre and post-processing). □

Proposition 13 ($B(2^\lambda+1)$-approximate correctness of refreshed evaluated ciphertexts)**.** *For all polynomials ν, δ, all $\lambda \in \mathbb{N}$, all crs containing a large enough modulus $N \in \mathbb{N}$, all messages $\mu \in \{0,1\}^{\nu(\lambda)}$, all circuits $f : \{0,1\}^{\nu(\lambda)} \to \mathbb{Z}_N$ of depth at most $\delta(\lambda)$, we have:* $\Pr\Big[(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(\mathsf{crs}), \mathsf{ct} \leftarrow \mathsf{Enc}_{\mathsf{pk}}(\mu), \mathsf{ct}_f = \mathsf{Eval}(\mathsf{pk}, f, 0, \mathsf{ct}), \mathsf{ct}'_f \leftarrow \mathsf{ReRand}(\mathsf{pk}, \mathsf{ct}_f) : |\mathsf{sk}^\top \mathsf{ct}'_f - f(\mu)| \leq B(2^\lambda + 1)\Big] \in 1 - 2^{-\Omega(\lambda)}$.

Proof: The ciphertext ct_f is the binary decomposition of $\left(\mathbf{A}(\mathbf{r}^\star + \mathbf{r}_f), (\mathbf{s}^\top\mathbf{A} + \mathbf{e}^\top)(\mathbf{r}^\star + \mathbf{r}_f) + f(\mu)\right) \in \mathbb{Z}_N^{\kappa+1}$, where $|\mathbf{e}^\top\mathbf{r}_f| < B$ by batch correctness and $|\mathbf{e}^\top\mathbf{r}^\star| < B_\chi B^\star m = 2^\lambda B$ with probability $1 - 2^{-\Omega(\lambda)}$ over the choices of $\mathbf{e} \leftarrow \chi^m$. □

5.2 XiO Construction

We directly dive into the formal description of the construction, see the introduction for a detailed overview.

We present a modular construction of XiO for the class of circuits $\mathcal{C}_{\log(n),s,d}$ for polynomials n, s, d, from the following building blocks:

- the GSW FHE scheme for depth δ circuits, denoted by $\mathrm{GSW}_\delta = (\mathsf{CRSgen}, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Enc}^\star, \mathsf{Dec}, \mathsf{Eval}, \mathsf{Eval}', \mathsf{Eval}_{\mathsf{rand}}, \mathsf{ReRand})$, presented in Section 3.2.2. The depth δ is chosen sufficiently large to handle the homomorphic evaluations of the circuits described below.

- an $(\ell_1, \ell_2, h, \alpha)$-Hintable Packed LHE, denoted by $\mathcal{LHE}_{b,n,\varepsilon} = (\overline{\mathsf{Gen}}, \overline{\mathsf{Enc}}, \overline{\mathsf{Enc}^\star}, \overline{\mathsf{Dec}}, \overline{\mathsf{Eval}}, \overline{\mathsf{SecHint}}, \overline{\mathsf{PubHint}}, \overline{\mathsf{Rec}}, \overline{\mathsf{VerKey}})$ where h is independent of n to ensure succinctness; $\ell_1(\lambda) \geq b(\lambda) + 2\lambda$, where 2^b is a bound on the noise obtained when FHE evaluating circuits of depth at most δ[12]; moreover $\big(\ell_1(\lambda) - b(\lambda) - 2\lambda\big) \cdot \ell_2(\lambda) \geq n^\varepsilon(\lambda)$, where $\varepsilon \in (0,1)$ is defined below (see the paragraph about succinctness); finally $\alpha(\lambda) \leq \lambda + 1$.

Notations. For every program Π with $\log(n)$ bits inputs, every $\varepsilon \in (0,1)$, the truth table can be written as $(\Pi_i)_{i \in [n^{1-\varepsilon}]}$, where each chunk Π_i contains n^ε bits. The chunks Π_i themselves can be subdivided into sub-chunks $\Pi_i = (\Pi_{i,j})_{j \in [\ell_2]}$, where each sub-chunk $\Pi_{i,j}$ contains n^ε/ℓ_2 bits. For all $i \in [n^{1-\varepsilon}]$ and $j \in [\ell_2]$, we denote by $C_{i,j}$ the circuit that takes as input a program Π of size s and outputs $\Pi_{i,j}$.

The construction: We proceed to the construction.

• $\underline{\mathsf{Gen}_{\mathsf{Obf}}(1^\lambda)}$:
Set the parameters:

- Choose a constant $0 < \varepsilon < 1$ that is small enough so as to ensure succinctness of the scheme (see paragraph succinctness below).
- Let $|\overline{\mathsf{ct}}|(\cdot)$, $|\overline{\mathbf{r}^\star}|(\cdot)$ be polynomials such that for every $\lambda \in \mathbb{N}$, every $(\overline{\mathsf{pk}}, \overline{\mathsf{sk}})$ in the support of $\overline{\mathsf{Gen}}(1^\lambda)$ that defines the message space $\mathbb{Z}_N^{\ell_2}$ and the noisy randomness space $\mathcal{R}^\star$, every message $\mathbf{m} \in \mathbb{Z}_N^{\ell_2}$, every ciphertext in the support of $\overline{\mathsf{Enc}}_{\overline{\mathsf{pk}}}(\mathbf{m})$ has a bit size at most $|\overline{\mathsf{ct}}|(\lambda)$ and every $\mathbf{r}^\star \in \mathcal{R}^\star$ has bit size at most $|\mathbf{r}^\star|(\lambda)$.
- $\mathsf{FHE.PubCoin} \leftarrow_{\mathsf{R}} \{0,1\}^{n^{1-\varepsilon} \cdot \ell_2 \cdot |\mathbf{r}^\star|}$, $\mathsf{LHE.PubCoin} \leftarrow_{\mathsf{R}} \{0,1\}^{n^{1-\varepsilon} \cdot |\overline{\mathsf{ct}}|}$.

Return $\mathsf{pp} = (\mathsf{FHE.PubCoin}, \mathsf{LHE.PubCoin})$.

• $\underline{\mathsf{Obf}(\mathsf{pp}, 1^n, \Pi)}$:
Sample the following parameters:

- $\big(\overline{\mathsf{pk}}, \overline{\mathsf{sk}}\big) \leftarrow \overline{\mathsf{Gen}}(1^\lambda)$ that defines the message space $\mathbb{Z}_N^{\ell_2}$.
- $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(\overline{\mathsf{pk}})$ that defines the noisy randomness space $\mathcal{R}^\star$, where $\mathsf{sk} \in \mathbb{Z}_N^w$, and pk contains the noise bound B; we write $b = \lceil \log(B) \rceil$.

Compute the following ciphertexts:

- $\mathsf{ct}_1 \leftarrow \mathsf{Enc}_{\mathsf{pk}}(\Pi)$.
- $\mathsf{ct}_2 \leftarrow \mathsf{Enc}_{\mathsf{pk}}(\overline{\mathsf{sk}})$.
- $\overline{\mathsf{ct}} \leftarrow \overline{\mathsf{Enc}}_{\overline{\mathsf{pk}}}(\mathsf{sk})$.

For all $i \in [n^{1-\varepsilon}]$, $j \in [\ell_2]$, compute the following:

- $\mathsf{ct}_{i,j} = \mathsf{Eval}'(\mathsf{pk}, C_{i,j}, b + 2\lambda, \mathsf{ct}_1) \in \{0,1\}^w$, where the circuit $C_{i,j}$ is defined above. The homomorphic evaluation is performed with scaling factor $b + 2\lambda$.

[12] To make sure these parameters are instantiable, we require that LHE decryption is of poly-logarithmic depth, which ensures that δ and therefore B only depend poly-logarithmically on ℓ_1.

- $\mathsf{ct}_{\mathsf{MSB},i,j} = \mathsf{Eval}'(\mathsf{pk}, f_{i,j}, 0, \mathsf{ct}_2) \in \{0,1\}^w$, where the circuit $f_{i,j}$ takes as input a bit string $\mathbf{a} \in \{0,1\}^{|\overline{\mathsf{sk}}|}$. It checks that $\mathbf{a}$ is the secret key associated with $\overline{\mathsf{pk}}$, that is, it runs $\overline{\mathsf{VerKey}}(\overline{\mathsf{pk}}, \mathbf{a})$. If the latter outputs 0, $f_{i,j}$ outputs 0. Otherwise, it uses $\mathbf{a}$ as an LHE secret key to compute $\mathbf{v}_i = \overline{\mathsf{Dec}}_{\mathbf{a}}(\mathsf{LHE.PubCoin}_i)$, where $\mathsf{LHE.PubCoin}_i$ is interpreted as an LHE ciphertext $\overline{\mathsf{Enc}}_{\overline{\mathsf{pk}}}(\mathbf{u}_i)$, with $\mathbf{u}_i \in \mathbb{Z}_N^{\ell_2}$, by density of the LHE ciphertext space. Then it computes $v_{i,j} \in \mathbb{Z}_N$, the j'th coordinate of $\mathbf{v}_i \in \mathbb{Z}_N^{\ell_2}$ and outputs the most significant bits of $v_{i,j}$, of the form: $\mathsf{MSB}(v_{i,j}) = v_{i,j} - \mathsf{LSB}(v_{i,j}) \in \mathbb{Z}_N^{\ell_2}$, where the (shifted) least significant bits are of the form: $\mathsf{LSB}(v_{i,j}) = v_{i,j} \bmod B2^{2\lambda} - B2^{2\lambda}/2 \in \mathbb{Z}_N$. The homomorphic evaluation is performed with scaling factor 1.
- Parse $\mathsf{FHE.PubCoin}_{i,j} = \mathbf{r}^\star_{i,j} \in \mathcal{R}^\star$ and compute $\mathsf{ct}'_{\mathsf{MSB},i,j} = \mathsf{ReRand}(\mathsf{pk}, \mathsf{ct}_{\mathsf{MSB},i,j}; \mathbf{r}^\star_{i,j}) \in \mathbb{Z}_N^{\kappa+1}$.
- Compute $\mathsf{ct}_i = (\mathsf{ct}_{i,j})_{j\in[\ell_2]} \in \{0,1\}^{w\ell_2}$, $\mathsf{ct}'_{\mathsf{MSB},i} = (\mathsf{ct}'_{\mathsf{MSB},i,j})_{j\in[\ell_2]} \in \{0,1\}^{w\ell_2}$.
- Compute $\overline{\mathsf{ct}}_i = \overline{\mathsf{Eval}}\big(\overline{\mathsf{pk}}, \overline{\mathsf{ct}}, \mathsf{LHE.PubCoin}_i, \mathsf{ct}_i - \mathsf{ct}'_{\mathsf{MSB},i}\big)$.
- Compute $\rho_i \leftarrow \overline{\mathsf{SecHint}}(\overline{\mathsf{sk}}, \overline{\mathsf{ct}}_i)$.

Return $\widetilde{\Pi} = \big(\mathsf{pk}, \overline{\mathsf{pk}}, \mathsf{ct}_1, \mathsf{ct}_2, \overline{\mathsf{ct}}, \{\rho_i\}_{i\in[n^{1-\varepsilon}]}\big)$.

- $\underline{\mathsf{Eval}(\mathsf{pp}, \widetilde{\Pi}, \mathbf{x})}$:

  - Let $i \in [n^{1-\varepsilon}]$ such that $\Pi(\mathbf{x})$ belongs to the i'th chunk of the truth table of Π. Compute $\overline{\mathsf{ct}}_i$ as described above.
  - Recover $m_i \leftarrow \overline{\mathsf{Rec}}(\overline{\mathsf{pk}}, \overline{\mathsf{ct}}_i, \rho_i)$.
  - Compute $m'_i = \lfloor 2^{-2\lambda}/B \cdot m_i \rceil$, which contains $\Pi(\mathbf{x})$.

  We now proceed to prove Theorem 5.3.

Succinctness. By h-succinctness of $\mathcal{LHE}_{b,n,\varepsilon}$, for all $i \in [n^{1-\varepsilon}]$, we have $|\rho_i| \le h(\lambda)$ for a polynomial h that is independent of n. The rest of the obfuscated circuit $\widetilde{\Pi}$ is of size $p(\lambda, n^\varepsilon, d)$, for a polynomial p that is independent of n and ε. Thus, there exists a constant $c \in \mathbb{N}$ (independent of ε) and a polynomial q (independent of n) such that $\big|\widetilde{\Pi}\big| \in n^{c\varepsilon}(\lambda) \cdot q(\lambda, d) + n^{1-\varepsilon}(\lambda) \cdot h(\lambda)$. For succinctness, we pick an appropriately small $0 < \varepsilon < 1/c$.

Correctness.

- By the batch correctness of the GSW scheme (Proposition 1), for all $i \in [n^{1-\varepsilon}]$ and $j \in [\ell_2]$, we have:
$$\mathsf{sk}^\top \mathsf{ct}_{i,j} = 2^{2\lambda} B \cdot \Pi_{i,j} + \mathsf{noise}_{i,j} \in \mathbb{Z}_N,$$
where $|\mathsf{noise}_{i,j}| < B$.
- By the density of the noisy ciphertexts of $\mathcal{LHE}_{b,n,\varepsilon}$, for all $i \in [n^{1-\varepsilon}]$, we have $\mathsf{LHE.PubCoin}_i = \overline{\mathsf{Enc}}_{\overline{\mathsf{pk}}}(\mathbf{u}_i)$ with $\mathbf{u}_i \in \mathbb{Z}_N^{\ell_2}$.
- By the $(2^\lambda+1)B$-approximate correctness of refreshed evaluated ciphertexts of the GSW scheme (Proposition 13), for all $i \in [n^{1-\varepsilon}]$ and $j \in [\ell_2]$, we have:
$$\mathsf{sk}^\top \mathsf{ct}'_{\mathsf{MSB},i,j} = \mathsf{MSB}(u_{i,j}) + \mathsf{noise}_{\mathsf{MSB},i,j} \in \mathbb{Z}_N,$$
where $|\mathsf{noise}_{\mathsf{MSB},i,j}| < (2^\lambda + 1)B$.

- By linear homomorphism of $\mathcal{LHE}_{b,n,\varepsilon}$ (Property 4.2), the ciphertext $\overline{\mathsf{ct}}_i$ is in the support of $\overline{\mathsf{Enc}}_{\overline{\mathsf{pk}}}(\mathbf{m}_i)$, $\mathbf{m}_i = (m_{i,j})_{j\in[\ell_2]}$ of the form:

$$m_{i,j} = B2^{2\lambda}\cdot\Pi_{i,j} + \mathsf{LSB}(u_{i,j}) + \mathsf{noise}_{i,j} + \mathsf{noise}_{\mathsf{MSB},i,j} \in \mathbb{Z}_N.$$

- By α-correctness of the secret hints of $\mathcal{LHE}_{b,n,\varepsilon}$ (Property 4.3), the evaluator of the obfuscated circuit recovers the message $\overline{\mathbf{m}_i} \in \mathbb{Z}_N^{\ell_2}$ such that $\|\overline{\mathbf{m}_i} - \mathbf{m}_i\|_\infty < 2^{\alpha(\lambda)}$. That is, for all $j \in [\ell_2]$, $\overline{m_{i,j}} = m_{i,j} + \overline{\mathsf{noise}}_{i,j} \in \mathbb{Z}_N$, where $|\overline{\mathsf{noise}}_{i,j}| < 2^{\alpha(\lambda)}$.

- With probability $1 - 2^{-\Omega(\lambda)}$ over the choice of $\mathbf{u}_i \leftarrow_{\mathsf{R}} \mathbb{Z}_N^{\ell_2}$, we have for all $j \in [\ell_2]$, $|\mathsf{LSB}(u_{i,j}) + \mathsf{noise}_{i,j} + \mathsf{noise}_{\mathsf{MSB},i} + \overline{\mathsf{noise}_{i,j}}| < B2^{2\lambda}/2$. Thus, $m'_{i,j} = \left\lfloor \overline{m_{i,j}} \cdot 2^{-2\lambda}/B \right\rceil = \Pi_{i,j}$ for all $j \in [\ell_2]$, and the evaluator outputs $\Pi(\mathbf{x})$.

5.3 IND Security from 2CIRC

2-Circular Security. We define the notion of 2-circular security w.r.t two encryption schemes $\mathcal{PKE}, \overline{\mathcal{PKE}}$ similarly than for 1-circular security (see Definition 2.8). This notion will not be used for our final theorem, but will serve to state some intermediary results. For our purposes, the key generation algorithm of $\mathcal{PKE}$ will be allowed to depend on the public key and the secret of $\overline{\mathcal{PKE}}$ (so they can operate over the same field). To enables this, we make use of the CRS: the CRS of $\mathcal{PKE}$ will be set to the public key and the secret key of $\overline{\mathcal{PKE}}$. 2-circular security requires indistinguishability of encryptions using $\mathcal{PKE}$ of any message $\mathbf{m}^0, \mathbf{m}^1$ in the presence of encrypted key cycle w.r.t. $\mathcal{PKE}$ and $\overline{\mathcal{PKE}}$, and some randomness leakage.

Definition 5.2 ($\mathcal{O}$-leakage resilient 2-circular security)**.** *We say that public-key encryption schemes $\mathcal{PKE} = (\mathsf{CRSgen}, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ and $\overline{\mathcal{PKE}} = (\overline{\mathsf{CRSgen}}, \overline{\mathsf{Gen}}, \overline{\mathsf{Enc}}, \overline{\mathsf{Dec}})$ are $\mathcal{O}$-leakage resilient 2-circular secure if for all stateful nuPPT adversaries $\mathcal{A}$, there exists some negligible function $\mu(\cdot)$ such that for all $\lambda \in \mathbb{N}$, $\Pr[\mathsf{Exp}^{\mathcal{PKE},\overline{\mathcal{PKE}}}_{\lambda,\mathcal{A}} = 1] \le 1/2 + \mu(\lambda)$, where the experiment $\mathsf{Exp}^{\mathcal{PKE},\overline{\mathcal{PKE}}}_{\lambda,\mathcal{A}}$ is defined as follows:*

$$\mathsf{Exp}^{\mathcal{PKE},\overline{\mathcal{PKE}}}_{\lambda,\mathcal{A}} = \left\{ \begin{array}{l} \overline{\mathsf{crs}} \leftarrow \overline{\mathsf{CRSgen}}(1^\lambda), (\overline{\mathsf{pk}}, \overline{\mathsf{sk}}) \leftarrow \overline{\mathsf{Gen}}(\overline{\mathsf{crs}}), \mathsf{crs} = (\overline{\mathsf{pk}}, \overline{\mathsf{sk}}), (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(\mathsf{crs}) \\ (\mathbf{m}_0, \mathbf{m}_1) \leftarrow \mathcal{A}(\overline{\mathsf{pk}}, \mathsf{pk}), b \leftarrow \{0,1\}, \mathbf{m}^\star = \overline{\mathsf{sk}} \| \mathbf{m}^b, \mathbf{r} \leftarrow_{\mathsf{R}} \{0,1\}^\infty, \mathsf{ct} = \mathsf{Enc}_{\mathsf{pk}}(\mathbf{m}^\star; \mathbf{r}) \\ \overline{\mathsf{ct}} \leftarrow \overline{\mathsf{Enc}}_{\overline{\mathsf{pk}}}(\mathsf{sk}), b' \leftarrow \mathcal{A}^{\mathcal{O}(\mathbf{m}^\star, \mathbf{r})}(\mathsf{ct}, \overline{\mathsf{ct}}) \\ \textit{Return 1 if } |\mathbf{m}^0| = |\mathbf{m}^1|,\ b' = b \textit{ and } \mathcal{O} \textit{ did not return } \perp\textit{; 0 otherwise.} \end{array} \right\}$$

We say that an FHE scheme $\mathcal{FHE}$ and a PKE scheme $\overline{\mathcal{PKE}}$ are 2-circular SRL secure *if they are $\mathcal{O}_{\mathsf{SRL}}$-leakage resilient 2-circular secure, where the oracle $\mathcal{O}_{\mathsf{SRL}}$ is given in Definition 3.3.*

We now state our theorem.

Theorem 5.3. *Assume that for all polynomials δ, b, all constants $\varepsilon \in (0,1)$, there exists a polynomial h s.t. for all polynomials n, there exist polynomials ℓ_1, ℓ_2, α and an $(\ell_1, \ell_2, h, \alpha)$-hintable packed LHE denoted by $\mathcal{LHE}_{b,\varepsilon,n}$ s.t. for all $\lambda \in \mathbb{N}$:*

- $\alpha(\lambda) \le \lambda + 1$
- $\ell_1(\lambda) \ge b(\lambda)$
- $\big(\ell_1(\lambda) - b(\lambda)\big) \cdot \ell_2(\lambda) > n(\lambda)^\varepsilon$

- *2-circular SRL security (resp. subexponential 2-circular SRL security) holds w.r.t.* $\mathcal{LHE}_{b,\varepsilon,n}$ *and* GSW_δ.

Then XiO (resp. subexponentially secure XiO) for $\mathsf{P}^{\log}/\mathsf{poly}$ *exists.*

Proof: We now prove that the XiO scheme presented in Section 5 is IND-secure, provided 2-circular SRL security (as per Definition 3.3) holds w.r.t. GSW_δ and $\mathcal{LHE}_{b,n,\varepsilon}$.

We proceed via a hybrid argument using the experiments described below for all $b \in \{0,1\}$ and all $\lambda \in \mathbb{N}$.

• $\underline{\mathcal{H}_\lambda^{b.0}}$: this is the experiment from Definition 2.3. For completeness, we describe it here.

- Generation of pp: for all $i \in [n^{1-\varepsilon}]$, $\mathsf{LHE.PubCoin}_i \leftarrow_R \{0,1\}^{|\overline{\mathsf{ct}}|}$, for all $j \in [\ell_2]$, $\mathsf{FHE.PubCoin}_{i,j} \leftarrow_R \mathcal{R}^\star$, where $\mathcal{R}^\star$ denotes the noisy randomness space of $\mathcal{FHE}$. Return $\mathsf{pp} = \Big((\mathsf{LHE.PubCoin}_i)_{i\in[n^{1-\varepsilon}]}, (\mathsf{FHE.PubCoin}_{i,j})_{i\in[n^{1-\varepsilon}],j\in[\ell_2]}\Big)$.
- Generation of $\widetilde{\Pi_b}$: $(\overline{\mathsf{pk}}, \overline{\mathsf{sk}}) \leftarrow \overline{\mathsf{Gen}}(1^\lambda)$, $\mathsf{crs} = (\overline{\mathsf{pk}}, \overline{\mathsf{sk}})$, $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(\mathsf{crs})$, $\mathsf{ct}_1 \leftarrow \mathsf{Enc}_{\mathsf{pk}}(\Pi)$, $\mathsf{ct}_2 \leftarrow \mathsf{Enc}_{\mathsf{pk}}(\overline{\mathsf{sk}})$, $\overline{\mathsf{ct}} \leftarrow \overline{\mathsf{Enc}}_{\overline{\mathsf{pk}}}(\mathsf{sk})$. For all $i \in [n^{1-\varepsilon}]$, $j \in [\ell_2]$, compute the following:
  - $\mathsf{ct}_{i,j} = \mathsf{Eval}'(\mathsf{pk}, C_{i,j}, b + 2\lambda, \mathsf{ct}_1) \in \{0,1\}^w$;
  - $\mathsf{ct}_{\mathsf{MSB},i,j} = \mathsf{Eval}'(\mathsf{pk}, f_{i,j}, 0, \mathsf{ct}_2) \in \{0,1\}^w$;
  - Parse $\mathsf{FHE.PubCoin}_{i,j} = \mathbf{r}^\star_{i,j} \in \mathcal{R}^\star$ and compute $\mathsf{ct}'_{\mathsf{MSB},i,j} = \mathsf{ReRand}(\mathsf{pk}, \mathsf{ct}_{\mathsf{MSB},i,j}; \mathbf{r}^\star_{i,j}) \in \{0,1\}^w$.
  - Compute $\mathsf{ct}_i = (\mathsf{ct}_{i,j})_{j\in[\ell_2]} \in \{0,1\}^{w\ell_2}$, $\mathsf{ct}'_{\mathsf{MSB},i} = (\mathsf{ct}'_{\mathsf{MSB},i,j})_{j\in[\ell_2]} \in \{0,1\}^{w\ell_2}$.
  - $\overline{\mathsf{ct}}_i = \overline{\mathsf{Eval}}(\overline{\mathsf{pk}}, \overline{\mathsf{ct}}, \mathsf{LHE.PubCoin}_i, \mathsf{ct}_i - \mathsf{ct}'_{\mathsf{MSB},i})$;
  - $\rho_i \leftarrow \overline{\mathsf{SecHint}}(\overline{\mathsf{sk}}, \overline{\mathsf{ct}}_i)$.

  Return $\widetilde{\Pi}_b = \big(\mathsf{pk}, \overline{\mathsf{pk}}, \mathsf{ct}_1, \mathsf{ct}_2, \overline{\mathsf{ct}}, (\rho_i)_{i\in[n^{1-\varepsilon}]}\big)$.

• $\underline{\mathcal{H}_\lambda^{b.1}}$: the experiment samples $\mathsf{LHE.PubCoin}$ as in $\mathcal{H}_\lambda^{b.0}$, but does not sample $\mathsf{FHE.PubCoin}$ just yet; it then generates $\widetilde{\Pi}_b$ as in $\mathcal{H}_\lambda^{b.0}$ up until the point that the $\mathsf{ct}_{\mathsf{MSB},i,j}$ get re-randomized into $\mathsf{ct}'_{\mathsf{MSB},i,j}$ via ReRand. Next, instead of performing the re-randomization, it samples $\mathsf{ct}'_{\mathsf{MSB},i,j}$ as a *fresh* extra noisy encryption of $\mathsf{MSB}(u_{i,j})$ using randomness $\mathbf{r}^\star_{i,j} \leftarrow_R \mathcal{R}^\star$, and setting $\mathsf{FHE.PubCoin}_{i,j}$ to be $\mathbf{r}^\star_{i,j} - \mathbf{r}_{f_{i,j}}$, where $\mathbf{r}_{f_{i,j}}$ denotes the evaluated randomness computed via $\mathsf{Eval}_{\mathsf{rand}}$. Afterwards, the experiment continues exactly the same way as in the experiment $\mathcal{H}_\lambda^{b.0}$.

We show that for all $b \in \{0,1\}$, we have:

$$\{\mathcal{H}_\lambda^{b.0}\}_{\lambda\in\mathbb{N}} \approx_s \{\mathcal{H}_\lambda^{b.1}\}_{\lambda\in\mathbb{N}},$$

using the weak circuit privacy of $\mathcal{FHE}$ (Theorem 5.1).

The latter states that for all $\lambda \in \mathbb{N}$, all $(\overline{\mathsf{pk}}, \overline{\mathsf{sk}})$ in the support of $\overline{\mathsf{Gen}}(1^\lambda)$, all $(\mathsf{pk}, \mathsf{sk})$ in the support of $\mathsf{Gen}(\mathsf{crs})$ with $\mathsf{crs} = (\overline{\mathsf{pk}}, \overline{\mathsf{sk}})$, for all depth d-circuits and in particular the functions $f_{i,j}$

defined previously, these two distributions have statistical distance at most $2^{-\Omega(\lambda)}$:

$$\mathcal{D}^0_\lambda : \left\{ \begin{array}{l} r \leftarrow_{\mathsf{R}} \left([-1,1]^{m\times w}\right)^{|\overline{\mathsf{sk}}|}, \mathsf{ct} = \mathsf{Enc}_{\mathsf{pk}}(\overline{\mathsf{sk}}; r), \forall i \in [n^{1-\varepsilon}], j \in [\ell_2], \mathsf{ct}_{f_{i,j}} = \mathsf{Eval}'(\mathsf{pk}, f_{i,j}, 0, \mathsf{ct}) \\ \mathbf{r}^\star_{i,j} \leftarrow_{\mathsf{R}} \mathcal{R}^\star, \mathsf{ct}^\star_{f_{i,j}} = \mathsf{ReRand}(\mathsf{pk}, \mathsf{ct}_{f_{i,j}}; \mathbf{r}^\star_{i,j}) : \left(\overline{\mathsf{pk}}, \mathsf{pk}, \mathsf{ct}, \left(\mathbf{r}^\star_{i,j}, \mathsf{ct}^\star_{f_{i,j}}\right)_{i\in[n^{1-\varepsilon}], j\in[\ell_2]}\right) \end{array} \right\}$$

$$\mathcal{D}^1_\lambda : \left\{ \begin{array}{l} r \leftarrow_{\mathsf{R}} \left([-1,1]^{m\times w}\right)^{|\overline{\mathsf{sk}}|}, \mathsf{ct} = \mathsf{Enc}_{\mathsf{pk}}(\overline{\mathsf{sk}}; r), \forall i \in [n^{1-\varepsilon}], j \in [\ell_2], \mathbf{r}^\star_{i,j} \leftarrow_{\mathsf{R}} \mathcal{R}^\star \\ \mathsf{ct}^\star_{f_{i,j}} = \mathsf{Enc}^\star_{\mathsf{pk}}(\mathsf{MSB}(u_{i,j}); \mathbf{r}^\star_{i,j}) \\ \mathbf{r}_{f_{i,j}} = \mathsf{Eval}_{\mathsf{rand}}\left(\mathsf{pk}, f_{i,j}, r, \overline{\mathsf{sk}}\right) : \left(\overline{\mathsf{pk}}, \mathsf{pk}, \mathsf{ct}, \left(\mathbf{r}^\star_{i,j} - \mathbf{r}_{f_{i,j}}, \mathsf{ct}^\star_{f_{i,j}}\right)_{i\in[n^{1-\varepsilon}], j\in[\ell_2]}\right) \end{array} \right\}.$$

We design an inefficient simulator $\mathcal{S}$ that given a tuple $\left(\overline{\mathsf{pk}}, \mathsf{pk}, \mathsf{ct}, (\mathbf{r}_{i,j}, \mathsf{ct}_{i,j})_{i\in[n^{1-\varepsilon}], j\in[\ell_2]}\right)$, simulates the adversary view in the XiO security experiment. That is, we show that when fed with an input distributed according to $\mathcal{D}^0_\lambda$, $\mathcal{S}$ simulates the experiment $\mathcal{H}^{b.0}_\lambda$, whereas it simulates the experiment $\mathcal{H}^{b.1}_\lambda$ when fed with an input distributed according to $\mathcal{D}^1_\lambda$.

Given $\left(\overline{\mathsf{pk}}, \mathsf{pk}, \mathsf{ct}, (\mathbf{r}_{i,j}, \mathsf{ct}^\star_{f_{i,j}})_{i\in[n^{1-\varepsilon}], j\in[\ell_2]}\right)$, $\mathcal{S}$ (inefficiently) recovers $\overline{\mathsf{sk}}$ from $\overline{\mathsf{pk}}$, sk from pk, and the randomness r from ct (more precisely $\mathcal{S}$ samples some uniformly random $\overline{\mathsf{sk}}$, sk, r among those that match $\overline{\mathsf{pk}}$, pk and ct). It samples $\mathsf{LHE.PubCoin} \leftarrow_{\mathsf{R}} \{0,1\}^{n^{1-\varepsilon}\cdot|\overline{\mathsf{ct}}|}$, and for all $i \in [n^{1-\varepsilon}]$, $j \in [\ell_2]$, sets $\mathsf{FHE.PubCoin}_{i,j} = \mathbf{r}_{i,j}$, and $\mathsf{pp} = (\mathsf{LHE.PubCoin}, (\mathsf{FHE.PubCoin}_{i,j})_{i,j})$. It computes $\mathsf{ct}_1 \leftarrow \mathsf{Enc}_{\mathsf{pk}}(\Pi_b)$, $\overline{\mathsf{ct}} \leftarrow \overline{\mathsf{Enc}}_{\overline{\mathsf{pk}}}(\mathsf{sk})$.

For all $i \in [n^{1-\varepsilon}]$, $j \in [\ell_2]$, $\mathcal{S}$ computes the following:

- $\mathsf{ct}_{i,j} = \mathsf{Eval}'(\mathsf{pk}, C_{i,j}, b + 2\lambda, \mathsf{ct}_1) \in \{0,1\}^w$;
- $\mathsf{ct}'_{\mathsf{MSB},i,j} = \mathsf{ct}_{i,j} \in \{0,1\}^w$.
- Compute $\mathsf{ct}_i = (\mathsf{ct}_{i,j})_{j\in[\ell_2]} \in \{0,1\}^{w\ell_2}$, $\mathsf{ct}'_{\mathsf{MSB},i} = (\mathsf{ct}'_{\mathsf{MSB},i,j})_{j\in[\ell_2]} \in \{0,1\}^{w\ell_2}$.
- $\overline{\mathsf{ct}}_i = \overline{\mathsf{Eval}}(\overline{\mathsf{pk}}, \overline{\mathsf{ct}}, \mathsf{LHE.PubCoin}_i, \mathsf{ct}_i - \mathsf{ct}'_{\mathsf{MSB},i})$;
- $\rho_i \leftarrow \overline{\mathsf{SecHint}}(\overline{\mathsf{sk}}, \overline{\mathsf{ct}}_i)$.

The simulator sets $\widetilde{\Pi}_b = \left(\mathsf{pk}, \overline{\mathsf{pk}}, \mathsf{ct}_1, \mathsf{ct}_2, \overline{\mathsf{ct}}, (\rho_i)_{i\in[n^{1-\varepsilon}]}\right)$, and returns $(\mathsf{pp}, \widetilde{\Pi}_b)$. It is clear from the description of the simulator $\mathcal{S}$ that when the latter is fed with an input distributed according to $\mathcal{D}^0_\lambda$, it simulates the experiment $\mathcal{H}^{b.0}_\lambda$, whereas it simulates the experiment $\mathcal{H}^{b.1}_\lambda$ when fed with an input distributed according to $\mathcal{D}^1_\lambda$.

• $\mathcal{H}^{b.2}_\lambda$: this experiment is the same as $\mathcal{H}^{b.1}_\lambda$, except that instead of sampling $\mathsf{LHE.PubCoin}_i$ as random strings, they are sampled as fresh LHE ciphertexts of random plaintexts, that is, of the form $\mathsf{LHE.PubCoin}_i \leftarrow \overline{\mathsf{Enc}}_{\overline{\mathsf{pk}}}(\mathbf{u}_i)$ for $\mathbf{u}_i \leftarrow_{\mathsf{R}} \mathbb{Z}^{\ell_2}_N$. By density of the ciphertexts of $\mathcal{LHE}$ Property 4.7, we have:

$$\{\mathcal{H}^{b.1}_\lambda\}_{\lambda\in\mathbb{N}} \approx_s \{\mathcal{H}^{b.2}_\lambda\}_{\lambda\in\mathbb{N}}.$$

• $\mathcal{H}^{b.3}_\lambda$: this experiment is the same as $\mathcal{H}^{b.2}_\lambda$, except the ciphertexts $\overline{\mathsf{ct}}_i$ are generated as fresh noisy LHE encryptions of the messages $\mathbf{m}_i = \mathsf{sk}^\top(\mathsf{ct}_i - \mathsf{ct}'_{\mathsf{MSB},i}) + \mathbf{u}_i \in \mathbb{Z}^{\ell_2}_N$, and the $\mathsf{LHE.PubCoin}_i$ are instead computed homomorphically by subtracting the LHE encryption of the message $\widetilde{\mathbf{m}}_i = \mathsf{sk}^\top(\mathsf{ct}_i + \mathsf{ct}'_{\mathsf{MSB},i}) \in \mathbb{Z}^{\ell_2}_N$ from the fresh noisy encryption of $\mathbf{m}_i$. That is, for all $i \in [n^{1-\varepsilon}]$, $\overline{\mathsf{ct}}_i \leftarrow \overline{\mathsf{Enc}}^\star_{\overline{\mathsf{pk}}}(\mathbf{m}_i)$, $\mathsf{LHE.PubCoin}_i = \overline{\mathsf{Eval}}\left(\overline{\mathsf{pk}}, \overline{\mathsf{ct}}, \overline{\mathsf{ct}}_i, -\mathsf{ct}_i + \mathsf{ct}'_{\mathsf{MSB},i}\right)$.

Note that it is possible to define this hybrid since $\mathsf{ct}'_{\mathsf{MSB},i}$ remains exactly the same no matter what $\mathsf{LHE.PubCoin}_i$ is. This was not true in $\mathcal{H}^{b.0}_\lambda$, and we introduced $\mathcal{H}^{b.1}_\lambda$ to break this dependency.

We show that for all $b \in \{0,1\}$, we have:

$$\{\mathcal{H}^{b.2}_\lambda\}_{\lambda\in\mathbb{N}} \approx_s \{\mathcal{H}^{b.3}_\lambda\}_{\lambda\in\mathbb{N}},$$

using the weak circuit privacy property of $\mathcal{LHE}_{b,n,\varepsilon}$ (Property 4.6).

The latter states that for all $\lambda \in \mathbb{N}$, all $(\overline{\mathsf{pk}}, \overline{\mathsf{sk}})$ in the support of $\overline{\mathsf{Gen}}(1^\lambda)$, all vectors $\mathbf{x} \in \mathbb{Z}_N^w$ and in particular $\mathbf{x} = \mathsf{sk} \in \mathbb{Z}_N^w$, all $\mathbf{u}_i \in \mathbb{Z}_N^{\ell_2}$, all functions $\mathbf{y}^i = (\mathbf{y}^i_1, \ldots, \mathbf{y}^i_{\ell_2}) \in [-1,1]^{w\ell_2}$ and in particular the vector $\mathsf{ct}_i - \mathsf{ct}'_{\mathsf{MSB},i} \in [-1,1]^{w\ell_2}$ defined previously for all $i \in [n^{1-\varepsilon}]$, the following distributions have statistical distance $2^{-\Omega(\lambda)}$:

$$\mathcal{D}^0_\lambda = \left\{ \begin{array}{l} \overline{\mathsf{ct}} \leftarrow \overline{\mathsf{Enc}}_{\overline{\mathsf{pk}}}(\mathsf{sk}), \forall i \in [n^{1-\varepsilon}], \widetilde{\mathsf{ct}_i} \leftarrow \overline{\mathsf{Enc}^\star_{\overline{\mathsf{pk}}}}(\mathbf{u}_i) \\ \overline{\mathsf{ct}}_i = \overline{\mathsf{Eval}}(\overline{\mathsf{pk}}, \overline{\mathsf{ct}}, \widetilde{\mathsf{ct}}_i, \mathsf{ct}_i - \mathsf{ct}'_{\mathsf{MSB},i}) : \left(\overline{\mathsf{pk}}, \overline{\mathsf{ct}}, (\widetilde{\mathsf{ct}}_i, \overline{\mathsf{ct}}_i)_{i\in[n^{1-\varepsilon}]}\right) \end{array} \right\}$$

$$\mathcal{D}^1_\lambda = \left\{ \begin{array}{l} \overline{\mathsf{ct}} \leftarrow \overline{\mathsf{Enc}}_{\overline{\mathsf{pk}}}(\mathsf{sk}), \forall i \in [n^{1-\varepsilon}], \overline{\mathsf{ct}}_i \leftarrow \overline{\mathsf{Enc}^\star_{\overline{\mathsf{pk}}}}(\mathbf{m}_i) \\ \widetilde{\mathsf{ct}_i} = \overline{\mathsf{Eval}}(\overline{\mathsf{pk}}, \overline{\mathsf{ct}}, \overline{\mathsf{ct}}_i, -\mathsf{ct}_i + \mathsf{ct}'_{\mathsf{MSB},i}) : \left(\overline{\mathsf{pk}}, \overline{\mathsf{ct}}, (\widetilde{\mathsf{ct}_i}, \overline{\mathsf{ct}}_i)_{i\in[n^{1-\varepsilon}]}\right) \end{array} \right\},$$

where for all $i \in [n^{1-\varepsilon}]$, $\mathbf{m}_i = \mathsf{sk}^\top(\mathsf{ct}_i - \mathsf{ct}'_{\mathsf{MSB},i}) + \mathbf{u}_i \in \mathbb{Z}_N^{\ell_2}$.

We design an inefficient simulator $\mathcal{S}$ that given a tuple $\left(\overline{\mathsf{pk}}, \overline{\mathsf{ct}}, (\widetilde{\mathsf{ct}}_i, \overline{\mathsf{ct}}_i)_{i\in[n^{1-\varepsilon}]}\right)$, simulates the adversary view in the XiO security experiment. That is, we show that when fed with an input distributed according to $\mathcal{D}^0_\lambda$, $\mathcal{S}$ simulates the experiment $\mathcal{H}^{b.2}_\lambda$, whereas it simulates the experiment $\mathcal{H}^{b.3}_\lambda$ when fed with an input distributed according to $\mathcal{D}^1_\lambda$.

Given $\left(\overline{\mathsf{pk}}, \overline{\mathsf{ct}}, (\widetilde{\mathsf{ct}}_i, \overline{\mathsf{ct}}_i)_{i\in[n^{1-\varepsilon}]}\right)$, $\mathcal{S}$ (inefficiently) recovers $\overline{\mathsf{sk}}$ from $\overline{\mathsf{pk}}$, sk from $\overline{\mathsf{ct}}$, pk from sk and $\mathbf{u}_i$ from $\widetilde{\mathsf{ct}}_i$ for all $i \in [n^{1-\varepsilon}]$. It generates $\mathsf{ct}_1 \leftarrow \mathsf{Enc}_{\mathsf{pk}}(\Pi_b)$, $r \leftarrow_R \left([-1,1]^{m\times w}\right)^{|\overline{\mathsf{sk}}|}$, $\mathsf{ct}_2 = \mathsf{Enc}_{\mathsf{pk}}(\overline{\mathsf{sk}}; r)$, for all $i \in [n^{1-\varepsilon}]$, $j \in [\ell_2]$, $\mathsf{ct}_{i,j} = \mathsf{Eval}'(\mathsf{pk}, C_{i,j}, b + 2\lambda, \mathsf{ct}_1)$, $\mathbf{r}^\star_{i,j} \leftarrow_R \mathcal{R}^\star$, where $\mathcal{R}^\star$ denotes the noisy randomness space of $\mathcal{FHE}_d$, $\mathsf{ct}'_{\mathsf{MSB},i,j} = \mathsf{Enc}^\star_{\mathsf{pk}}(\mathsf{MSB}(u_{i,j}); \mathbf{r}^\star_{i,j})$, $\mathbf{r}_{f_{i,j}} = \mathsf{Eval}_{\mathsf{rand}}(\mathsf{pk}, f_{i,j}, r, \overline{\mathsf{sk}})$, $\mathsf{FHE.PubCoin}_{i,j} = \mathbf{r}^\star_{i,j} - \mathbf{r}_{f_{i,j}}$, $\rho_i \leftarrow \overline{\mathsf{SecHint}}(\overline{\mathsf{sk}}, \overline{\mathsf{ct}}_i)$, $\mathsf{LHE.PubCoin}_i = \widetilde{\mathsf{ct}}_i$.

It returns $\mathsf{pp} = \left((\mathsf{LHE.PubCoin}_i)_i, (\mathsf{FHE.PubCoin}_{i,j})_{i,j}\right)$ and $\widetilde{\Pi_b} = \left(\mathsf{pk}, \overline{\mathsf{pk}}, \mathsf{ct}_1, \mathsf{ct}_2, \overline{\mathsf{ct}}, (\rho_i)_{i\in[n^{1-\varepsilon}]}\right)$. It is clear from the description of the simulator $\mathcal{S}$ that when the latter is fed with an input distributed according to $\mathcal{D}^0_\lambda$, it simulates $\mathcal{H}^{b.2}_\lambda$, whereas it simulates $\mathcal{H}^{b.3}_\lambda$ when fed with an input distributed according to $\mathcal{D}^1_\lambda$.

• $\underline{\mathcal{H}^{b.4}_\lambda}$: it is the same experiment as $\mathcal{H}^{b.3}_\lambda$, except the hints ρ_i for $i \in [n^{1-\varepsilon}]$ are computed using $\overline{\mathsf{PubHint}}(\overline{\mathsf{pk}}, r_i)$, where r_i denotes the randomness used to produce the ciphertext $\overline{\mathsf{ct}}_i$; instead of $\overline{\mathsf{SecHint}}(\overline{\mathsf{sk}}, \overline{\mathsf{ct}}_i)$. By Property 4.4 of the LHE, we have $\{\mathcal{H}^{b.3}_\lambda\}_{\lambda\in\mathbb{N}} \approx_s \{\mathcal{H}^{b.4}_\lambda\}_{\lambda\in\mathbb{N}}$. Note that in the experiment $\mathcal{H}^{b.4}_\lambda$, we no longer use the LHE secret key $\overline{\mathsf{sk}}$.

• $\underline{\mathcal{H}^{b.5}_\lambda}$: it is the same experiment as $\mathcal{H}^{b.4}_\lambda$, except that the ciphertexts $\overline{\mathsf{ct}}_i$ are generated as a fresh encryption of a message $\mathbf{m}_i \in \mathbb{Z}_N^{\ell_2}$ of the form $(m_{i,1}, \ldots, m_{i,\ell_2})$ where for all $j \in [\ell_2]$, we have $m_{i,j} = B2^{2\lambda} \cdot \Pi^b_{i,j} + \mathsf{LSB}(u_{i,j}) \in \mathbb{Z}_N$. Recall that $\mathsf{LSB}(u_{i,j}) = u_{i,j} \mod B2^{2\lambda} - B2^{2\lambda}/2 \in \mathbb{Z}_N$. This is instead of having $m_{i,j} = \mathsf{sk}^\top(\mathsf{ct}_{i,j} + \mathsf{ct}'_{\mathsf{MSB},i,j}) + u_{i,j} = B2^{2\lambda} \cdot \Pi^b_{i,j} + \mathsf{LSB}(u_{i,j}) + \mathsf{noise}_{i,j} + \mathsf{noise}_{\mathsf{MSB},i} \in \mathbb{Z}_N$, where $\mathsf{noise}_{i,j} = \mathbf{e}^\top \mathbf{r}_{C_{i,j}} \in \mathbb{Z}_N$ and $\mathsf{noise}_{\mathsf{MSB},i} = \mathbf{e}^\top \mathbf{r}^\star_{i,j} \in \mathbb{Z}_N$, $\mathbf{r}_{C_{i,j}}$ is the randomness obtained when evaluating the circuit $C_{i,j}$ on the FHE ciphertext ct_1, $\mathbf{r}^\star_{i,j} \leftarrow_R \mathcal{R}^\star$, and $\mathbf{e} \leftarrow \chi^m$ is used to generate pk.

Note that $\mathsf{noise}_{i,j}$ and $\mathsf{noise}_{\mathsf{MSB},i}$ are deterministic functions of pk, $\mathbf{r}^\star_{i,j}$ and the randomness $r \in \left([-1,1]^{m\times w}\right)^s$ used to produce ct_1. In particular, they are independent of $\mathsf{LSB}(u_{i,j})$.

We show that $\{\mathcal{H}^{b.3}_\lambda\}_{\lambda\in\mathbb{N}} \approx_s \{\mathcal{H}^{b.4}_\lambda\}_{\lambda\in\mathbb{N}}$. To do so, for all $\lambda \in \mathbb{N}$, we exhibit two distributions $\mathcal{D}^0_\lambda$ and $\mathcal{D}^1_\lambda$, together with a (possibly inefficient) simulator $\mathcal{S}$, such that (1) $\mathcal{D}^0_\lambda$ and $\mathcal{D}^1_\lambda$ have statistical distance $2^{-\Omega(\lambda)}$, and (2) for all $\beta \in \{0,1\}$, when fed with an input from distribution $\mathcal{D}^\beta_\lambda$, $\mathcal{S}$ produces the adversary view as in the experiment $\mathcal{H}^{b.4+\beta}_\lambda$.

The distributions are defined as follows (the differences are highlighted in red):

$$\mathcal{D}^0_\lambda = \left\{ \begin{array}{c} (\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{Gen}(\mathsf{crs}), r \leftarrow_{\mathsf{R}} \left([-1,1]^{m\times w}\right)^s, \forall i \in [n^{1-\varepsilon}], j \in [\ell_2], \mathbf{r}^\star_{i,j} \leftarrow_{\mathsf{R}} \mathcal{R}^\star \\ \gamma_{i,j} \leftarrow_{\mathsf{R}} \left[-B2^{2\lambda}/2+1, B2^{2\lambda}/2\right] : \left(\mathsf{pk}, r, \left(\gamma_{i,j}, \mathbf{r}^\star_{i,j}\right)_{i\in[n^{1-\varepsilon}],j\in[\ell_2]}\right) \end{array} \right\}$$

$$\mathcal{D}^1_\lambda = \left\{ \begin{array}{c} (\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{Gen}(\mathsf{crs}), r \leftarrow_{\mathsf{R}} \left([-1,1]^{m\times w}\right)^s, \forall i \in [n^{1-\varepsilon}], j \in [\ell_2], \mathbf{r}^\star_{i,j} \leftarrow_{\mathsf{R}} \mathcal{R}^\star \\ \gamma_{i,j} \leftarrow_{\mathsf{R}} \left[-B2^{2\lambda}/2+1, B2^{2\lambda}/2\right] : \left(\mathsf{pk}, r, \left(\gamma_{i,j} + \mathsf{noise}_{i,j} + \mathsf{noise}_{\mathsf{MSB},i,j}, \mathbf{r}^\star_{i,j}\right)_{i\in[n^{1-\varepsilon}],j\in[\ell_2]}\right) \end{array} \right\},$$

where $\mathsf{noise}_{i,j}$ $\mathsf{noise}_{\mathsf{MSB},i,j}$ are functions of pk, r and $\mathbf{r}^\star_{i,j}$ defined as below, namely, $\mathsf{noise}_{i,j} = \mathbf{e}^\top \mathbf{r}_{C_{i,j}}$ and $\mathsf{noise}_{\mathsf{MSB},i,j} = \mathbf{e}^\top \mathbf{r}^\star_{i,j}$.

We show that these distributions have statistical distance $2^{-\Omega(\lambda)}$. The only difference is that in $\mathcal{D}^1_\lambda$, an extra noise $\mathsf{noise}_{i,j} + \mathsf{noise}_{\mathsf{MSB},i,j}$ is added to the random value $s_{i,j}$. This noise is small, indeed $|\mathsf{noise}_{i,j} + \mathsf{noise}_{\mathsf{MSB},i,j}| \le B(2^\lambda+1)$ (see the correctness section for more details). Moreover, $\gamma_{i,j}$ is sampled uniformly at random over $\left[-B2^{2\lambda}/2+1, B2^{2\lambda}/2\right]$, independently of the other values output by the distributions. Thus, we can use the value $\gamma_{i,j}$ to smudge the noise $\mathsf{noise}_{i,j} + \mathsf{noise}_{\mathsf{MSB},i,j}$. That is, by Lemma 2.2 (smudging), the statistical distance of the two distributions is $2^{-\Omega(\lambda)}$.

Now, we proceed to describe the simulator $\mathcal{S}$. Given as input the tuple $\left(\mathsf{pk}, r, (v_{i,j}, \mathbf{r}^\star_{i,j})_{i\in[n^{1-\varepsilon}],j\in[\ell_2]}\right)$, the simulator (inefficiently) recovers sk from pk, samples $(\overline{\mathsf{pk}}, \overline{\mathsf{sk}}) \leftarrow \overline{\mathsf{Gen}}(1^\lambda)$, generates $\overline{\mathsf{ct}} \leftarrow \overline{\mathsf{Enc}}_{\overline{\mathsf{pk}}}(\mathsf{sk})$, $\mathsf{ct}_1 = \mathsf{Enc}_{\mathsf{pk}}(\Pi_b; r)$. It samples $r' \leftarrow_{\mathsf{R}} \left([-1,1]^{m\times w}\right)^{|\overline{\mathsf{sk}}|}$, computes $\mathsf{ct}_2 = \mathsf{Enc}_{\mathsf{pk}}\left(\overline{\mathsf{sk}}; r'\right)$.

For all $i \in [n^{1-\varepsilon}]$, $j \in [\ell_2]$, samples $\omega_{i,j} \leftarrow_{\mathsf{R}} \mathbb{Z}_{N/(2^{2\lambda}B)}$, and sets $m_{i,j} = B2^{2\lambda}\cdot\Pi^b_{i,j} + v_{i,j} + \omega_{i,j} \in \mathbb{Z}_N$. Note that $(\gamma_{i,j}, \omega_{i,j})$ is identically distributed to $(\mathsf{LSB}(u_{i,j}), \mathsf{MSB}(u_{i,j}))$ for $u_{i,j} \leftarrow_{\mathsf{R}} \mathbb{Z}_N$.

It sets $\mathbf{m}_i = (m_{i,1}, \ldots, m_{i,\ell_2}) \in \mathbb{Z}^{\ell_2}_N$, samples $r_i \leftarrow_{\mathsf{R}} \overline{\mathcal{R}}$, where $\overline{\mathcal{R}}$ denotes the randomness space of $\overline{\mathsf{Enc}}$, computes $\overline{\mathsf{ct}}_i = \overline{\mathsf{Enc}}_{\overline{\mathsf{pk}}}(\mathbf{m}_i; r_i)$ and $\rho_i \leftarrow \overline{\mathsf{PubHint}}(\overline{\mathsf{pk}}, r_i)$. It computes $\mathsf{ct}'_{\mathsf{MSB},i,j} = \mathsf{Enc}^\star_{\mathsf{pk}}(\omega_{i,j}; \mathbf{r}^\star_{i,j})$, $\mathsf{ct}_{i,j} = \mathsf{Eval}'(\mathsf{pk}, C_{i,j}, b+2\lambda, \mathsf{ct}_1)$, $\mathsf{ct}_i = (\mathsf{ct}_{i,j})_{j\in[\ell_2]} \in \{0,1\}^{w\ell_2}$, $\mathsf{ct}'_{\mathsf{MSB},i} = (\mathsf{ct}'_{\mathsf{MSB},i,j})_{j\in[\ell_2]} \in \{0,1\}^{w\ell_2}$, and $\mathsf{LHE.PubCoin}_i = \overline{\mathsf{Eval}}(\overline{\mathsf{pk}}, \overline{\mathsf{ct}}, \overline{\mathsf{ct}}_i, -\mathsf{ct}_i + \mathsf{ct}'_{\mathsf{MSB},i})$. It computes $\mathbf{r}_{f_{i,j}} = \mathsf{Eval}_{\mathsf{rand}}(\mathsf{pk}, f_{i,j}, \mathsf{ct}_2)$ where the functions $f_{i,j}$ are defined as before, and sets $\mathsf{FHE.PubCoin}_{i,j} = \mathbf{r}^\star_{i,j} - \mathbf{r}_{f_{i,j}} \in \mathbb{Z}^m_N$.

It returns $\mathsf{pp} = (\mathsf{LHE.PubCoin}_i, \mathsf{FHE.PubCoin}_{i,j})_{i\in[n^{1-\varepsilon}]}$, and $\widetilde{\Pi}_b = \left(\mathsf{pk}, \overline{\mathsf{pk}}, \mathsf{ct}_1, \mathsf{ct}_2, \overline{\mathsf{ct}}, (\rho_i)_{i\in[n^{1-\varepsilon}]}\right)$. When $\mathcal{S}$ is fed with an input distributed according to $\mathcal{D}^0_\lambda$, it simulates the experiment $\mathcal{H}^{b.4}_\lambda$, whereas it simulates the experiment $\mathcal{H}^{b.5}_\lambda$ when fed with distribution $\mathcal{D}^1_\lambda$. Thus, we have:

$$\{\mathcal{H}^{b.4}_\lambda\}_{\lambda\in\mathbb{N}} \approx_s \{\mathcal{H}^{b.5}_\lambda\}_{\lambda\in\mathbb{N}}.$$

• $\underline{\{\mathcal{H}^{0.5}_\lambda\}_{\lambda\in\mathbb{N}} \approx_c \{\mathcal{H}^{1.5}_\lambda\}_{\lambda\in\mathbb{N}}}$: To complete the proof, we show that $\{\mathcal{H}^{0.5}_\lambda\}_{\lambda\in\mathbb{N}}$ is computationally indistinguishable from $\{\mathcal{H}^{1.5}_\lambda\}_{\lambda\in\mathbb{N}}$. These two ensembles are the same except the former obfuscates the program Π^0, whereas the latter obfuscates Π^1. Note that other than the encrypted key cycle, we never use the FHE secret key, and due to hybrid $\{\mathcal{H}^{b.4}_\lambda\}_{\lambda\in\mathbb{N}}$, we no longer use the LHE secret key. The coins $\mathsf{FHE.PubCoin}$ exactly correspond to an SRL leakage on the FHE ciphertext ct_2 (and note that in the experiment we do know the output $\alpha_{i,j}$ of the function $f_{i,j}$ that is applied to the plaintexts encrypted in $\mathsf{ct}_1, \mathsf{ct}_2$—namely, it is $\mathsf{MSB}(u_{i,j})$ where $u_{i,j}$ is a random element of $\mathbb{Z}_N$ selected in the experiment, see Hybrid $\mathcal{H}^{b.2}_\lambda$). Thus, (subexponential) indistinguishability of $\{\mathcal{H}^{0.5}_\lambda\}_{\lambda\in\mathbb{N}}$ and $\{\mathcal{H}^{1.5}_\lambda\}_{\lambda\in\mathbb{N}}$ follows from (subexponential) 2-circular SRL-security of $\mathcal{FHE}_d$ and $\mathcal{LHE}_{b,n,\varepsilon}$.

Namely, for any nuPPT (resp. subexponential) distinguisher $\mathcal{A}$, we show there exists a nuPPT (resp. subexponential) reduction $\mathcal{B}$ such that for all $\lambda \in \mathbb{N}$, we have:

$$\Pr\left[b \leftarrow_{\mathsf{R}} \{0,1\}, (\mathsf{pp}, \widetilde{\Pi}^b) \leftarrow \mathcal{H}_\lambda^{b.5} : \mathcal{A}(\mathsf{pp}, \widetilde{\Pi}^b) = b\right] \leq \Pr\left[\mathsf{Exp}_{\lambda,\mathcal{B}}^{\mathcal{FHE}_d, \mathcal{LHE}_{b,n,\varepsilon}} = 1\right] + 2^{-\Omega(\lambda)},$$

where the experiment $\mathsf{Exp}_{\lambda,\mathcal{B}}^{\mathcal{FHE}_d, \mathcal{LHE}_{b,n,\varepsilon}}$ is described in Definition 5.2.

We now proceed to describe the reduction $\mathcal{B}$. Given the public keys pk, $\overline{\mathsf{pk}}$, $\mathcal{B}$ sends the pair (Π^0, Π^1) to the 2-circular SRL security experiment, upon which it receives the ciphertexts $\mathsf{ct} = (\mathsf{ct}_1 \| \mathsf{ct}_2)$ and $\overline{\mathsf{ct}}$, where ct_1 encrypts Π^0 or Π^1, ct_2 encrypts $\overline{\mathsf{sk}}$, and $\overline{\mathsf{ct}}$ encrypts sk. It samples $u_{i,j} \leftarrow_{\mathsf{R}} \mathbb{Z}_N$ for all $i \in [n^{1-\varepsilon}]$, $j \in [\ell_2]$, and generates the following:

- Generation of pp:
  - To generate $\mathsf{LHE.PubCoin}_i$:
    - It samples $r_i \leftarrow_{\mathsf{R}} \overline{\mathcal{R}}$, where $\overline{\mathcal{R}}$ denotes the randomness space of $\overline{\mathsf{Enc}}$, and computes $\overline{\mathsf{ct}}_i = \overline{\mathsf{Enc}}_{\overline{\mathsf{pk}}}(\mathbf{m}_i; r_i)$, where for all $j \in [\ell_2]$, the j'th coordinate of $\mathbf{m}_i$ is of the form: $m_{i,j} = B 2^{2\lambda} \cdot \Pi^b_{i,j} + \mathsf{LSB}(u_{i,j}) \in \mathbb{Z}_N$. Note that this does not require to know the bit b, since $\Pi^0_{i,j} = \Pi^1_{i,j}$ for all $i \in [n^{1-\varepsilon}]$, $j \in [\ell_2]$, because the programs Π^0 and Π^1 are functionally equivalent.
    - It computes $\rho_i \leftarrow \overline{\mathsf{PubHint}}(\overline{\mathsf{pk}}, r_i)$.
    - It computes $\mathsf{ct}_{i,j} = \mathsf{Eval}'(\mathsf{pk}, C_{i,j}, b + 2\lambda, \mathsf{ct}_1)$.
    - Then, it queries its $\mathcal{O}_{\mathsf{SRL}}$ oracle, to obtain a fresh, extra noisy encryption $\mathsf{Enc}^\star_{\mathsf{pk}}(0; \mathbf{r}^\star_{i,j})$. It leaves the oracle $\mathcal{O}_{\mathsf{SRL}}$ pending.
    - It adds the vector $(\mathbf{0}, \mathsf{MSB}(u_{i,j})) \in \mathbb{Z}_N^{\kappa+1}$ to the vector whose binary decomposition is $\mathsf{Enc}^\star_{\mathsf{pk}}(0; \mathbf{r}^\star_{i,j})$, which yields a vector $\mathsf{ct}^\star_{i,j} \in \mathbb{Z}_N^{\kappa+1}$ whose binary decomposition is $\mathsf{Enc}^\star_{\mathsf{pk}}(\mathsf{MSB}(u_{i,j}); \mathbf{r}^\star_{i,j})$. Then, it computes $\mathsf{ct}'_{\mathsf{MSB},i,j} = \mathsf{BD}(\mathsf{ct}^\star_{i,j}) \in \{0,1\}^w$.
    - It computes $\mathsf{ct}_i = (\mathsf{ct}_{i,j})_{j \in [\ell_2]} \in \{0,1\}^{w\ell_2}$ and $\mathsf{ct}'_{\mathsf{MSB},i} = (\mathsf{ct}'_{\mathsf{MSB},i,j})_{j \in [\ell_2]} \in \{0,1\}^{w\ell_2}$.
    - Finally, it computes $\mathsf{LHE.PubCoin}_i = \overline{\mathsf{Eval}}(\overline{\mathsf{pk}}, \overline{\mathsf{ct}}, \overline{\mathsf{ct}}_i, -\mathsf{ct}_i - \mathsf{ct}'_{\mathsf{MSB},i})$.
  - To generate $\mathsf{FHE.PubCoin}_{i,j}$: it answers the pending oracle $\mathcal{O}_{\mathsf{SRL}}$ with the function $f_{i,j}$ and the value $\alpha = \mathsf{MSB}(u_{i,j})$. With probability $1 - 2^{\Omega(\lambda)}$ over the random coins used to produce The oracle $\mathcal{O}_{\mathsf{SRL}}$ returns the leakage $\mathbf{r}^\star_{i,j} - \mathbf{r}_{f_{i,j}} \in \mathcal{R}^\star$. The reduction sets $\mathsf{FHE.PubCoin}_{i,j} = \mathbf{r}^\star_{i,j} - \mathbf{r}_{f_{i,j}}$.

  It sets $\mathsf{pp} = \big((\mathsf{LHE.PubCoin}_i)_{i \in [n^{1-\varepsilon}]}, (\mathsf{FHE.PubCoin}_{i,j})_{i \in [n^{1-\varepsilon}], j \in [\ell_2]}\big)$.
- Generation of $\widetilde{\Pi}$: it sets $(\mathsf{pk}, \overline{\mathsf{pk}}, \overline{\mathsf{ct}}, \mathsf{ct}, (\rho_i)_{i \in [n^{1-\varepsilon}])}$ computed as described above.

The reduction $\mathcal{B}$ sends $(\mathsf{pp}, \widetilde{\Pi})$ to the distinguisher $\mathcal{A}$, which outputs a bit b'. Finally, $\mathcal{B}$ outputs the bit b'. When ct_1 encrypts Π^0, the reduction simulates the experiment $\mathcal{H}_\lambda^{0.5}$ to $\mathcal{A}$, whereas it simulates the experiment $\mathcal{H}_\lambda^{1.5}$ when ct_1 encrypts Π^1. Thus, we have $\Pr\left[\mathsf{Exp}_{\lambda,\mathcal{B}}^{\mathcal{FHE}_d, \mathcal{LHE}_{b,n,\varepsilon}} = 1\right] \geq \Pr\left[b \leftarrow_{\mathsf{R}} \{0,1\}, (\mathsf{pp}, \widetilde{\Pi}^b) \leftarrow \mathcal{H}_\lambda^{b.5} : \mathcal{A}(\mathsf{pp}, \widetilde{\Pi}^b) = b\right]$. Overall, we have shown that:

$$\{\mathcal{H}_\lambda^{0.1}\}_{\lambda \in \mathbb{N}} \approx_s \{\mathcal{H}_\lambda^{0.2}\}_{\lambda \in \mathbb{N}} \approx_s \{\mathcal{H}_\lambda^{0.3}\}_{\lambda \in \mathbb{N}} \approx_s \{\mathcal{H}_\lambda^{0.4}\}_{\lambda \in \mathbb{N}} \approx_s \{\mathcal{H}_\lambda^{0.5}\}_{\lambda \in \mathbb{N}} \approx_c$$
$$\{\mathcal{H}_\lambda^{1.5}\}_{\lambda \in \mathbb{N}} \approx_s \{\mathcal{H}_\lambda^{1.4}\}_{\lambda \in \mathbb{N}} \approx_s \{\mathcal{H}_\lambda^{1.3}\}_{\lambda \in \mathbb{N}} \approx_s \{\mathcal{H}_\lambda^{1.2}\}_{\lambda \in \mathbb{N}} \approx_s \{\mathcal{H}_\lambda^{1.1}\}_{\lambda \in \mathbb{N}}.$$

□

5.4 XiO from 1-circular Security of GSW

We finally show how to base security on just the LWE assumption and the $\mathsf{1CIRC}^{\mathcal{O}_{\mathsf{SRL}}}$-assumption w.r.t. GSW— i.e. the assumption that SRL security of GSW is preserved in the presence of an encryption of the GSW secret key.

We do this in two steps. First, we remark that in our $Xi\mathcal{O}$ construction, security still holds if both the FHE and LHE use the same secret key (as long as the 2-circularly security of the two schemes holds in this setting). To that end, we describe a variant of the GSW FHE, denoted by GSW$'$ (Section 5.4.2), where the secret key is given as part of the CRS (instead of being sampled by the key generation algorithm).

Next, we present a slight modification of Packed-Regev, called Packed-Regev', where the secret key $\mathbf{s}$ is just a vector like in GSW and we then expand it into a Packed-Regev secret key (which is a matrix) by tensoring with the identify matrix. Moreover, the public key of Packed-Regev$'$ LHE uses an LWE noise of larger magnitude than Packed-Regev LHE. This way, the LHE public key can be generated from a GSW public key (recall the random self reducibility of LWE increases the noise magnitude). Taking larger noises only strengthens security of the scheme. The magnitude is taken to be negligible to all other smudging-size noises used by the scheme, so all properties that require smudging the noise used in the public key are still satisfied. That is, Packed-Regev' is a proper hintable LHE as per Definition 4.1. Its description is provided in Section 5.4.1.

We finally remark that, as is well known for the Regev scheme, 1-circular security directly holds also for Packed-Regev$'$. Thus, 2-circular SRL security of GSW and Packed-Regev' is implied by just 1-circular security of GSW. This is proven in Section 5.4.3.

5.4.1 Packed-Regev$'$

We now introduce Packed-Regev$'$ which is identical to Packed-Regev except for the distribution of the secret key, the dimensions of the public matrix $\mathbf{A}$, and the magnitude of the LWE noises used. Namely, the secret key is now a vector of the form $\mathbf{s} \leftarrow \chi^\kappa$, and it is used to generate a matrix $\mathbf{S}$ of the form $\mathbf{S} = \mathbf{s}^\top \otimes \mathrm{Id}_{\ell_2} = \begin{pmatrix} \mathbf{s}^\top & 0 & \cdots \\ 0 & \mathbf{s}^\top & \\ \vdots & & \ddots \end{pmatrix} \in \mathbb{Z}_N^{\ell_2 \times \ell_2 \kappa}$. The matrix $\mathbf{A} \in \mathbb{Z}_N^{\ell_2 w \times m}$ (instead of dimensions $\kappa \times m$ in Packed-Regev). Finally, the LWE noise $\mathbf{E}$ is sampled as $\mathbf{E} \leftarrow_{\mathsf{R}} [-2^{\lambda/3}, 2^{\lambda/3}]^{\ell_2 \times m}$, instead of the polynomially-bounded distribution $\chi^{\ell_2 \times m}$. This will be useful to generate the LHE public key from a GSW public key (which is needed to reduce 1-circular security of GSW to correlated-key 2-circular security).

Note that this is still an LHE since: none of the properties relied on the distribution of $\mathbf{S}$; the properties that relied on smudging the noise $\mathbf{E}$ are still satisfied, since the magnitude of $\mathbf{E}$ — $2^{\lambda/3}$ — is still negligible compared to other smudging size noises (of magnitude $2^{\lambda/2}$ and 2^λ for normal and noisy encryptions, respectively). It also satisfies semantic security, because even though $\mathbf{S}$ is sparse, the public key $\mathbf{SA} + \mathbf{E}$ still contains of a bunch of LWE samples, of the form $\mathbf{SA} + \mathbf{E} = \begin{pmatrix} \mathbf{s}^\top \mathbf{A}_1 + \mathbf{e}_1^\top \\ \vdots \\ \mathbf{s}^\top \mathbf{A}_{\ell_2} + \mathbf{e}_{\ell_2}^\top \end{pmatrix}$, where $\mathbf{A} = \begin{pmatrix} \mathbf{A}_1 \\ \vdots \\ \mathbf{A}_{\ell_2} \end{pmatrix}$, $\mathbf{A}_i \in \mathbb{Z}_N^{\kappa \times m}$ and $\mathbf{e}_i^\top \in \mathbb{Z}_N^{1 \times m}$ is the the i'th row of $\mathbf{E}$ for all $i \in [\ell_2]$. Thus, the public key can be turned to uniformly random using the LWE assumption (the fact that the noises $\mathbf{e}_i$ are now of larger magnitude can only make the LWE assumption weaker), and the same security proof that for Packed-Regev applies.

We now describe the Packed-Regev$'$ LHE in more details (the difference with the Packed-Regev LHE from Section 4.2 are highlighted in red). As for Packed-Regev, this scheme is parameterized by polynomials ℓ_1 and ℓ_2; we denote it by $\mathsf{P\text{-}Regev}'_{\ell_1,\ell_2}$.

• Gen(crs):
Given as input $\mathsf{crs} = 1^\lambda$, it chooses the parameters N, χ, κ exactly as described in Section 4.2. It sets $m = 2\ell_2\kappa\log(N) + 2\lambda$ (recall that in Packed-Regev, we have $m = 2\kappa\log(N) + 2\lambda$). It computes $(\mathbf{A}, T_\mathbf{A}) \leftarrow \mathsf{TrapGen}(1^\lambda, N, \ell_2\kappa)$ (recall that in Packed-Regev, the dimension of $\mathbf{A}$ is $\kappa \times m$),

$$\mathbf{s} \leftarrow_\mathsf{R} \chi^\kappa, \mathbf{S} = \mathbf{s}^\top \otimes \mathrm{Id}_{\ell_2} = \begin{pmatrix} \mathbf{s}^\top & 0 & \cdots \\ 0 & \mathbf{s}^\top & \\ \vdots & & \ddots \end{pmatrix} \in \mathbb{Z}_N^{\ell_2 \times \kappa\ell_2},$$

$\mathbf{E} \leftarrow [-2^{\lambda/3}, 2^{\lambda/3}]^{\ell_2 \times m}$, and sets $\mathsf{pk} = (N, \chi, \mathbf{A}, \mathbf{SA} + \mathbf{E})$, $\mathsf{sk} = \mathbf{s}$, $\mathsf{td} = T_\mathbf{A}$. It outputs $(\mathsf{pk}, \mathsf{sk}, \mathsf{td})$.

The PPT algorithms CRSgen, Enc, Enc*, Eval, Dec, SecHint, PubHint, Rec, VerKey are exactly the same as described in Section 4.2.

5.4.2 GSW$'$

We describe an FHE scheme which is exactly the GSW FHE described in Section 3.2.2, except the Gen algorithm, given as input a crs, checks that the crs contains an LWE secret $\mathbf{s}$, and uses it as its secret key (if crs doesn't contain such a vector, the algorithm aborts). The public key is computed from $\mathbf{s}$ as in the original GSW FHE. This minor change will allow us to consider an LHE scheme (namely, Packed-Regev$'$ described above) using *the same secret key* as the GSW$'$ encryption scheme. This scheme, as for the original GSW, is parameterized by a polynomial δ, and is denoted by GSW'_δ.

• Gen(crs):
Given as input crs which contains a modulus N and vector $\mathbf{s} \in \mathbb{Z}^\kappa$ and the description of a distribution χ over $\mathbb{Z}$ from a B_χ-bounded ensemble for a polynomial B_χ, it sets $\mathbf{g} = (1, 2, \ldots, 2^{\lceil\log(N)\rceil - 1}) \in \mathbb{Z}_N^{\lceil\log(N)\rceil}$, $w = (\kappa+1)\lceil\log(N)\rceil$ and $\mathsf{sk} = (-\mathbf{s}, 1) \otimes \mathbf{g} \in \mathbb{Z}_N^w$. The rest of the parameters is generated as in the original GSW presented in Section 3.2.2. Namely, it sets $m = 2(\kappa+1)\lceil\log(N)\rceil + 2\lambda$, $B^\star = 2^\lambda(w+1)^\delta\lceil\log(N)\rceil$ and $B = B_\chi(w+1)^\delta\lceil\log(N)\rceil m$. It samples $\mathbf{A} \leftarrow_\mathsf{R} \mathbb{Z}_N^{\kappa\times m}$, $\mathbf{e} \leftarrow \chi^m$, $\mathbf{G} = \mathbf{g}^\top \otimes \mathrm{Id}_{\kappa+1} \in \mathbb{Z}_N^{(\kappa+1)\times w}$, $\mathbf{U} = \begin{pmatrix} \mathbf{A} \\ \mathbf{s}^\top\mathbf{A} + \mathbf{e}^\top \end{pmatrix} \in \mathbb{Z}_N^{(\kappa+1)\times m}$ and sets $\mathsf{pk} = (B, \mathbf{U}, \mathbf{G})$. It outputs $(\mathsf{pk}, \mathsf{sk})$.

The algorithms Enc, Enc*, Eval, Dec, Eval$'$, $\mathsf{Eval}_\mathsf{rand}$, ReRand are defined exactly as for the GSW_δ scheme, given in Section 3.2.2 and in Section 5.1.

5.4.3 2-circular SRL Security from 1-circular SRL Security

Theorem 5.4. *Assume that for all polynomials δ, GSW_δ is (subexponentially) 1-circular SRL secure. Then for all polynomials δ, ℓ_1, ℓ_2, GSW'_δ and $\mathsf{P\text{-}Regev}'_{\ell_1,\ell_2}$ are (subexponentially) 2-circular SRL secure.*

Proof: The proof proceeds using the hybrid experiments listed below, defined for all $\lambda \in \mathbb{N}$.

• $\mathcal{H}^0_\lambda$: this is the experiment from Definition 5.2. For completeness, we describe it here. We write

$\mathsf{GSW}'_\delta = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Enc}^\star, \mathsf{Eval}, \mathsf{Eval}', \mathsf{Eval}_{\mathsf{rand}}, \mathsf{ReRand})$, and $\mathsf{P\text{-}Regev}'_{\ell_1,\ell_2} = (\overline{\mathsf{Gen}}, \overline{\mathsf{Enc}}, \overline{\mathsf{Enc}^\star}, \overline{\mathsf{Eval}}, \overline{\mathsf{SecHint}}, \overline{\mathsf{PubHint}}, \overline{\mathsf{Rec}}, \overline{\mathsf{Dec}}, \overline{\mathsf{VerKey}})$. The experiment $\mathcal{H}^0_\lambda$ generates the following:

- $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(\overline{\mathsf{pk}}, \overline{\mathsf{sk}})$, where $\mathsf{sk} = (-\mathbf{s}, 1) \otimes \mathbf{g} \in \mathbb{Z}_N^w$ with $w = (\kappa+1)\lceil \log(N) \rceil$ and $\mathsf{pk} = (B, \mathbf{U}, \mathbf{G})$ with $\mathbf{U} = \begin{pmatrix} \mathbf{A} \\ \mathbf{s}^\top \mathbf{A} + \mathbf{e}^\top \end{pmatrix} \in \mathbb{Z}_N^{(\kappa+1)\times m}$.
- $(\overline{\mathsf{pk}}, \overline{\mathsf{sk}}) \leftarrow \overline{\mathsf{Gen}}(1^\lambda)$, where $\overline{\mathsf{pk}} = \left(N, \chi, \overline{\mathbf{A}} \in \mathbb{Z}_N^{\kappa\ell_2 \times \overline{m}}, \left(\mathbf{s}^\top \otimes \mathrm{Id}_{\ell_2}\right) \overline{\mathbf{A}} + \overline{\mathbf{E}} \in \mathbb{Z}_N^{\ell_2 \times \overline{m}}\right)$ with $\overline{m} = 2\kappa\ell_2\lceil \log(N) \rceil + 2\lambda$, and $\overline{\mathsf{sk}} = \mathbf{s} \in \mathbb{Z}^\kappa$.

It sends the pair $(\overline{\mathsf{pk}}, \mathsf{pk})$ to the adversary, which returns a pair of messages $(\mathbf{m}^0, \mathbf{m}^1)$. The experiment then samples $b \leftarrow \{0,1\}$, $r \leftarrow_{\mathsf{R}} \left([-1,1]^{m\times w}\right)^{\kappa+|\mathbf{m}^b|}$, computes $\mathsf{ct} = \mathsf{Enc}_{\mathsf{pk}}(\overline{\mathsf{sk}} \| \mathbf{m}^b; r)$, $\overline{\mathsf{ct}} \leftarrow \overline{\mathsf{Enc}}_{\overline{\mathsf{pk}}}(\mathsf{sk})$ and sends $(\mathsf{pk}, \overline{\mathsf{pk}}, \mathsf{ct}, \overline{\mathsf{ct}})$ to the adversary, which also has an access to $\mathcal{O}_{\mathsf{SRL}}(\mathbf{m}^b, r)$. The adversary sins if it guesses b successfully and never makes the oracle $\mathcal{O}_{\mathsf{SRL}}(\mathbf{m}^b, r)$ output $\perp$.

• $\underline{\mathcal{H}^1_\lambda}$: this experiment is the same as $\mathcal{H}^0_\lambda$, except the challenge ciphertext $\overline{\mathsf{ct}}$ is computed as follows: $\overline{\mathsf{ct}} = \left(\overline{\mathbf{A}}\mathbf{R}, \left(\left(\mathbf{s}^\top \otimes \mathrm{Id}_{\ell_2}\right)\overline{\mathbf{A}}\mathbf{R} + \mathbf{E}' + \mathsf{sk}^\top \otimes \mathrm{Id}_{\ell_2}\right)\right.$ instead of $\overline{\mathsf{ct}} = \left(\overline{\mathbf{A}}\mathbf{R}, \left(\left(\left(\mathbf{s}^\top \otimes \mathrm{Id}_{\ell_2}\right)\overline{\mathbf{A}} + \overline{\mathbf{E}}\right)\mathbf{R} + \mathbf{E}' + \right.\right.$ $\left.\left.\mathsf{sk}^\top \otimes \mathrm{Id}_{\ell_2}\right)\right.$ in $\mathcal{H}^0_\lambda$ (the difference is highlighted in red). We show that

$$\{\mathcal{H}^0_\lambda\}_{\lambda\in\mathbb{N}} \approx_s \{\mathcal{H}^1_\lambda\}_{\lambda\in\mathbb{N}}.$$

By Lemma 2.2 (smudging), for all $\mathbf{R} \in [-1,1]^{\overline{m}\times\kappa\ell_2}$ and all $\overline{\mathbf{E}} \in [-2^{\lambda/3}, 2^{\lambda/3}]^{\ell_2\times\overline{m}}$, the following distributions have statistical distance $2^{-\Omega(\lambda)}$: $\{\mathbf{E}' \leftarrow_{\mathsf{R}} [-2^{\lambda/2}, 2^{\lambda/2}]^{\ell_2\times w\ell_2} : \mathbf{E}' + \overline{\mathbf{E}}\mathbf{R}\}$ and $\{\mathbf{E}' \leftarrow_{\mathsf{R}} [-2^{\lambda/2}, 2^{\lambda/2}]^{\ell_2\times w\ell_2} : \mathbf{E}'\}$. The first distribution (with pre and post-processing) corresponds to the experiment $\mathcal{H}^0_\lambda$, whereas the second distribution (with the same pre and post-processing) corresponds to $\mathcal{H}^1_\lambda$.

• $\underline{\mathcal{H}^2_\lambda}$: this experiment is the same as $\mathcal{H}^1_\lambda$, except the challenge ciphertext $\overline{\mathsf{ct}}$ uses $\mathbf{U} \leftarrow_{\mathsf{R}} \mathbb{Z}_N^{\kappa\ell_2\times w\ell_2}$, instead of $\overline{\mathbf{A}}\mathbf{R}$ with $\overline{\mathbf{A}} \leftarrow_{\mathsf{R}} \mathbb{Z}_N^{\kappa\ell_2\times\overline{m}}$ and $\mathbf{R} \leftarrow_{\mathsf{R}} [-1,1]^{\overline{m}\times w\ell_2}$. We prove that:

$$\{\mathcal{H}^1_\lambda\}_{\lambda\in\mathbb{N}} \approx_s \{\mathcal{H}^2_\lambda\}_{\lambda\in\mathbb{N}}.$$

To do so, we use Lemma 2.1 (leftover hash lemma), which states that the following distributions have statistical distance $2^{-\Omega(\lambda)}$: $\{\overline{\mathbf{A}} \leftarrow_{\mathsf{R}} \mathbb{Z}_N^{\kappa\ell_2\times\overline{m}}, \mathbf{R} \leftarrow_{\mathsf{R}} [-1,1]^{\overline{m}\times w\ell_2} : (\overline{\mathbf{A}}, \overline{\mathbf{A}}\mathbf{R})\}$ and $\{\overline{\mathbf{A}} \leftarrow_{\mathsf{R}} \mathbb{Z}_N^{\ell_2 w\times\overline{m}}, \mathbf{U} \leftarrow_{\mathsf{R}} \mathbb{Z}_N^{\kappa\ell_2\times w\ell_2} : (\overline{\mathbf{A}}, \mathbf{U})\}$. The first distribution corresponds (with pre and post-processing) to the experiment $\mathcal{H}^1_\lambda$, whereas the second distribution (with the same pre and post-processing) corresponds to $\mathcal{H}^2_\lambda$.

• $\underline{\mathcal{H}^3_\lambda}$: this experiment is the same as $\mathcal{H}^2_\lambda$, except the challenge ciphertext $\overline{\mathsf{ct}}$ uses $\mathbf{U} - \mathbf{g}^\top \otimes \mathrm{Id}_{\kappa\ell_2}$ where $\mathbf{U} \leftarrow_{\mathsf{R}} \mathbb{Z}_N^{\kappa\ell_2\times w\ell_2}$ and $\mathrm{Id}_{\kappa\ell_2} \in \mathbb{Z}_N^{\kappa\ell_2\times\kappa\ell_2}$ denotes the identity matrix, instead of $\mathbf{U}$. The two experiments $\mathcal{H}^2_\lambda$ and $\mathcal{H}^3_\lambda$ are the same, since the following are identically distributed: $\{\mathbf{U} \leftarrow_{\mathsf{R}} \mathbb{Z}_N^{\kappa\ell_2\times w\ell_2} : \mathbf{U}\}$ and $\{\mathbf{U} \leftarrow_{\mathsf{R}} \mathbb{Z}_N^{\kappa\ell_2\times w\ell_2} : \mathbf{U} - \mathbf{g}^\top \otimes \mathrm{Id}_{\kappa\ell_2}\}$. Note that in hybrid $\mathcal{H}^3_\lambda$, the ciphertext $\overline{\mathsf{ct}}$ is of the form: $\overline{\mathsf{ct}} = \left(\mathbf{U} - \mathrm{Id}_{\kappa\ell_2}, \left(\mathbf{s}^\top \otimes \mathrm{Id}_{\ell_2}\right)\mathbf{U} + \mathbf{E}'\right)$. To see this, note that $\mathbf{s}^\top \otimes \mathrm{Id}_{\ell_2} = \begin{pmatrix} \mathbf{s}^\top & & \\ & \ddots & \\ & & \mathbf{s}^\top \end{pmatrix} \in \mathbb{Z}_N^{\kappa\ell_2}$,

$\mathbf{g}^\top \otimes \mathrm{Id}_{\kappa\ell_2} = \begin{pmatrix} \mathbf{g}^\top & & \\ & \ddots & \\ & & \mathbf{g}^\top \end{pmatrix}$, and $\left(\mathbf{s}^\top \otimes \mathrm{Id}_{\ell_2}\right) \cdot \left(\mathbf{g}^\top \otimes \mathrm{Id}_{\kappa\ell_2}\right) = \begin{pmatrix} (\mathbf{s}\otimes\mathbf{g})^\top & & \\ & \ddots & \\ & & (\mathbf{s}\otimes\mathbf{g})^\top \end{pmatrix} =$

$(\mathbf{s} \otimes \mathbf{g})^\top \otimes \mathrm{Id}_{\ell_2}$.

• $\underline{\mathcal{H}^4_\lambda}$: this experiment is the same as $\mathcal{H}^3_\lambda$, except the matrix $\mathbf{U}$ is computed as follows: $\mathbf{U} = \begin{pmatrix} \mathbf{U}_1 \\ \vdots \\ \mathbf{U}_{\ell_2} \end{pmatrix}$ where for all $i \in [\ell_2]$, $\mathbf{U}_i = \mathbf{A}\mathbf{R}_i$ with $\mathbf{R}_i \leftarrow_{\mathsf{R}} [-1,1]^{m \times \ell_2}$. We prove that:

$$\{\mathcal{H}^3_\lambda\}_{\lambda \in \mathbb{N}} \approx_s \{\mathcal{H}^4_\lambda\}_{\lambda \in \mathbb{N}}.$$

To do so, we use the leftover hash lemma (Lemma 2.1) that states the following distributions have statistical distance $2^{-\Omega(\lambda)}$: $\{\mathbf{A} \leftarrow_{\mathsf{R}} \mathbb{Z}_N^{\kappa \times m}, \forall i \in [\ell_2], \mathbf{R}_i \leftarrow_{\mathsf{R}} [-1,1]^{m \times w\ell_2} : \left(\mathbf{A}, (\mathbf{A}\mathbf{R}_i)_{i \in [\ell_2]}\right)\}$ and $\{\mathbf{A} \leftarrow_{\mathsf{R}} \mathbb{Z}_N^{\kappa \times m}, \forall i \in [\ell_2], \mathbf{U}_i \leftarrow_{\mathsf{R}} \mathbb{Z}_N^{\kappa \times \kappa\ell_2} : \left(\mathbf{A}, (\mathbf{U})_{i \in [\ell_2]}\right)\}$. The first distribution (with pre and post-processing) corresponds to the experiment $\mathcal{H}^4_\lambda$, whereas the second distribution (with the same pre and post-processing) corresponds to $\mathcal{H}^3_\lambda$.

Note that in $\mathcal{H}^4_\lambda$, the ciphertext $\overline{\mathsf{ct}}$ is of the form: $\overline{\mathsf{ct}} = \left(\mathbf{U} - \mathbf{g}^\top \otimes \mathrm{Id}_{\kappa\ell_2}, \begin{pmatrix} \mathbf{s}^\top \mathbf{A}\mathbf{R}_1 \\ \vdots \\ \mathbf{s}^\top \mathbf{A}\mathbf{R}_{\ell_2} \end{pmatrix} + \mathbf{E}'\right)$.

• $\underline{\mathcal{H}^5_\lambda}$: this experiment is the same as $\mathcal{H}^4_\lambda$, except the challenge ciphertext $\overline{\mathsf{ct}}$ is as follows (the differences with previous hybrid highlighted in red): $\overline{\mathsf{ct}} = \left(\mathbf{U} - \mathbf{g}^\top \otimes \mathrm{Id}_{\kappa\ell_2}, \begin{pmatrix} (\mathbf{s}^\top \mathbf{A} + \mathbf{e}^\top)\mathbf{R}_1 \\ \vdots \\ (\mathbf{s}^\top \mathbf{A} + \mathbf{e}^\top)\mathbf{R}_{\ell_2} \end{pmatrix} + \mathbf{E}'\right)$. We show that

$$\{\mathcal{H}^4_\lambda\}_{\lambda \in \mathbb{N}} \approx_s \{\mathcal{H}^5_\lambda\}_{\lambda \in \mathbb{N}},$$

with statistical distance $2^{-\Omega(\lambda)}$. For all $\mathbf{e} \in \mathbb{Z}_N^m$ such that $\|\mathbf{e}\|_\infty \leq B_\chi$ for a polynomial B_χ and all $\mathbf{R}_i \in [-1,1]^{m \times w\ell_2}$, by Lemma 2.2, the following distributions have statistical distance $2^{-\Omega(\lambda)}$: $\{\mathbf{E}' \leftarrow_{\mathsf{R}} [-2^{\lambda/2}, 2^{\lambda/2}]^{\ell_2 \times \kappa\ell_2} : \mathbf{E}'\}$ and $\{\mathbf{E}' \leftarrow_{\mathsf{R}} [-2^{\lambda/2}, 2^{\lambda/2}]^{\ell_2 \times \kappa\ell_2} : \mathbf{E}' + \begin{pmatrix} \mathbf{e}^\top \mathbf{R}_1 \\ \vdots \\ \mathbf{e}^\top \mathbf{R}_{\ell_2} \end{pmatrix}\}$. The first distribution (with pre and post-processing) corresponds to the experiment $\mathcal{H}^4_\lambda$, whereas the second distribution (with the same pre and post-processing) corresponds to $\mathcal{H}^5_\lambda$.

• $\underline{\mathcal{H}^6_\lambda}$: this experiments is the same as $\mathcal{H}^5_\lambda$, except $\overline{\mathsf{pk}}$ is computed as follows: for all $i \in [\ell_2]$, $\overline{\mathbf{R}}_i \leftarrow_{\mathsf{R}} [-1,1]^{m \times \overline{m}}$, $\overline{\mathbf{A}} = \begin{pmatrix} \mathbf{A}\overline{\mathbf{R}}_1 \\ \vdots \\ \mathbf{A}\overline{\mathbf{R}}_{\ell_2} \end{pmatrix} \in \mathbb{Z}_N^{\kappa\ell_2 \times \overline{m}}$. By the leftover hash lemma (Lemma 2.1) with pre and post-processing, we have:

$$\{\mathcal{H}^5_\lambda\}_{\lambda \in \mathbb{N}} \approx_s \{\mathcal{H}^6_\lambda\}_{\lambda \in \mathbb{N}}.$$

Note that in hybrid $\mathcal{H}^6_\lambda$, the public key $\overline{\mathsf{pk}}$ is of the form: $\overline{\mathsf{pk}} = \left(\overline{\mathbf{A}}, \begin{pmatrix} \mathbf{s}^\top \mathbf{A}\overline{\mathbf{R}}_1 \\ \vdots \\ \mathbf{s}^\top \mathbf{A}\overline{\mathbf{R}}_{\ell_2} \end{pmatrix} + \overline{\mathbf{E}}\right)$.

• $\underline{\mathcal{H}^7_\lambda}$: this experiment is the same as $\mathcal{H}^6_\lambda$, except $\overline{\mathsf{pk}}$ is of the form (the differences with previous

hybrid highlighted in red): $\overline{\mathsf{pk}} = \left(\overline{\mathbf{A}}, \begin{pmatrix} (\mathbf{s}^\top \mathbf{A} + \mathbf{e}^\top)\overline{\mathbf{R}}_1 \\ \vdots \\ (\mathbf{s}^\top \mathbf{A} + \mathbf{e}^\top)\overline{\mathbf{R}}_{\ell_2} \end{pmatrix} + \overline{\mathbf{E}}\right)$. We prove that:

$$\{\mathcal{H}^6_\lambda\}_{\lambda\in\mathbb{N}} \approx_s \{\mathcal{H}^7_\lambda\}_{\lambda\in\mathbb{N}},$$

with statistical distance $2^{-\Omega(\lambda)}$. To do so, we use the fact that for all $\mathbf{e} \in \mathbb{Z}_N^m$ such that $\|\mathbf{e}\|_\infty \leq B_\chi$ for a polynomial B_χ, ll $\overline{\mathbf{R}}_i \in [-1,1]^{m\times\kappa\ell_2}$, by Lemma 2.2 (smudging), the following distribution have statistical distance $2^{-\Omega(\lambda)}$: $\{\overline{\mathbf{E}} \leftarrow_{\mathsf{R}} [-2^{\lambda/3}, 2^{\lambda/3}]^{\ell_2\times\overline{m}} : \overline{\mathbf{E}}\}$ and $\{\overline{\mathbf{E}} \leftarrow_{\mathsf{R}} [-2^{\lambda/3}, 2^{\lambda/3}]^{\ell_2\times\overline{m}} : \overline{\mathbf{E}} + \begin{pmatrix} \mathbf{e}^\top\overline{\mathbf{R}}_1 \\ \vdots \\ \mathbf{e}^\top\overline{\mathbf{R}}_{\ell_2} \end{pmatrix}\}$. The first distributions (with pre and post-processing) corresponds to the experiment $\mathcal{H}^6_\lambda$, whereas the second distribution (with the same pre and post-processing) corresponds to $\mathcal{H}^7_\lambda$.

Finally, we prove that for all nuPPT adversaries $\mathcal{A}$ playing against the experiment $\mathcal{H}^7_\lambda$, there exists an efficient reduction $\mathcal{B}$ that can break the 1-circular SRL security of GSW_δ with an advantage as large as the advantage of $\mathcal{A}$. That is,

Given as input the $\mathsf{pk} = (B, \mathbf{A}, \mathbf{s}^\top\mathbf{A}+\mathbf{e}^\top, \mathbf{G})$ from its experiment, the reduction $\mathcal{B}$ samples $\overline{\mathbf{R}}_i \leftarrow_{\mathsf{R}} [-1,1]^{m\times\overline{m}}$ for all $i \in [\ell_2]$, it $\overline{\mathbf{E}} \leftarrow_{\mathsf{R}} [-2^{\lambda/3}, 2^{\lambda/3}]^{\ell_2\times\overline{m}}$, $\overline{\mathbf{A}} = \begin{pmatrix} \mathbf{A}\overline{\mathbf{R}}_1 \\ \vdots \\ \mathbf{A}\overline{\mathbf{R}}_{\ell_2} \end{pmatrix}$, $\overline{\mathsf{pk}} = \left(\overline{\mathbf{A}}, \begin{pmatrix} (\mathbf{s}^\top\mathbf{A} + \mathbf{e}^\top)\overline{\mathbf{R}}_1 \\ \vdots \\ (\mathbf{s}^\top\mathbf{A} + \mathbf{e}^\top)\overline{\mathbf{R}}_{\ell_2} \end{pmatrix} + \overline{\mathbf{E}}\right)$, where again $(\mathbf{s}^\top\mathbf{A} + \mathbf{e}^\top)$ is taken from pk. It sends $(\mathsf{pk}, \overline{\mathsf{pk}})$ to $\mathcal{A}$, which answers with a pair of messages $(\mathbf{m}^0, \mathbf{m}^1)$. The reductions $\mathcal{B}$ forwards the pair $(\mathbf{m}^0, \mathbf{m}^1)$ to its experiment, upon which it receives the ciphertext $\mathsf{ct} = \mathsf{Enc}_{\mathsf{pk}}(\mathbf{s}\|\mathbf{m}^b)$ from its experiment. Then, for all $i \in [\ell_2]$, it samples $\mathbf{R}_i \leftarrow_{\mathsf{R}} [-1,1]^{m\times\kappa\ell_2}$, $\mathbf{E}' \leftarrow_{\mathsf{R}} [-2^{\lambda/2}, 2^{\lambda/2}]^{\ell_2\times\kappa\ell_2}$, $\overline{\mathsf{ct}} = \left(\begin{pmatrix} \mathbf{A}\mathbf{R}_1 \\ \vdots \\ \mathbf{A}\mathbf{R}_{\ell_2} \end{pmatrix} - \mathrm{Id}_{\kappa\ell_2}, \begin{pmatrix} (\mathbf{s}^\top\mathbf{A} + \mathbf{e}^\top)\mathbf{R}_1 \\ \vdots \\ (\mathbf{s}^\top\mathbf{A} + \mathbf{e}^\top)\mathbf{R}_{\ell_2} \end{pmatrix} + \mathbf{E}'\right)$, where $(\mathbf{s}^\top\mathbf{A} + \mathbf{e}^\top)$ is taken from pk. It sends $(\mathsf{ct}, \overline{\mathsf{ct}})$ to $\mathcal{A}$. Whenever $\mathcal{A}$ makes a query to its SRL oracle, $\mathcal{B}$ forwards the query to its own SRL oracle and sends back the answer to $\mathcal{A}$. □

5.5 Concluding Our Main Theorem

Combining Theorem 3.5 (SRL security of GSW from LWE), with Theorem 5.4 (1-circular SRL security of GSW ⇒ 2-circular SRL security of GSW' and Packed-Regev') and Theorem 5.3 (2-circular SRL security of GSW' and Packed-Regev' ⇒ XiO) we obtain the following theorem.

Theorem 5.5. *Assume the (subexponential) LWE assumption holds. Assume further that for all polynomial δ, $\mathsf{1CIRC}^{\mathcal{O}_{\mathsf{SRL}}}$ holds w.r.t GSW_δ. Then (subexponentially-secure) XiO for $\mathsf{P}^{\log}/\mathsf{poly}$ exists.*

Finally, combining with Theorem 2.4 (subexponential XiO and LWE ⇒ iO), we obtain our main theorem.

Theorem 5.6. *Assume the subexponential LWE assumption holds. Assume further that that for all polynomial δ, $\mathsf{1CIRC}^{\mathcal{O}_{\mathsf{SRL}}}$ holds w.r.t GSW_δ. Then subexponentially-secure iO for P/poly exists.*

Acknowledgments

We wish to thank Hoeteck Wee and Daniel Wichs for their insightful feedback on a previous eprint version of this paper. In particular, they encouraged us to present a game-based definition of SRL security (as opposed to the indistinguishability based definition we presented in our earlier draft), and to clarify differences between circular SRL-security and "plain" circular security.

References

[ABBC10] T. Acar, M. Belenkiy, M. Bellare, and D. Cash. Cryptographic agility and its relation to circular encryption. In *EUROCRYPT 2010*, *LNCS* 6110, pages 403–422. Springer, Heidelberg, May / June 2010.

[ACPS09] B. Applebaum, D. Cash, C. Peikert, and A. Sahai. Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In *CRYPTO 2009*, *LNCS* 5677, pages 595–618. Springer, Heidelberg, August 2009.

[Agr19] S. Agrawal. Indistinguishability obfuscation without multilinear maps: New methods for bootstrapping and instantiation. In *EUROCRYPT 2019, Part I*, *LNCS* 11476, pages 191–225. Springer, Heidelberg, May 2019.

[AJ15] P. Ananth and A. Jain. Indistinguishability obfuscation from compact functional encryption. In *CRYPTO 2015, Part I*, *LNCS* 9215, pages 308–326. Springer, Heidelberg, August 2015.

[AJKS18] P. Ananth, A. Jain, D. Khurana, and A. Sahai. Indistinguishability obfuscation without multilinear maps: iO from LWE, bilinear maps, and weak pseudorandomness. Cryptology ePrint Archive, Report 2018/615, 2018. https://eprint.iacr.org/2018/615.

[AJL+12] G. Asharov, A. Jain, A. López-Alt, E. Tromer, V. Vaikuntanathan, and D. Wichs. Multiparty computation with low communication, computation and interaction via threshold FHE. In *EUROCRYPT 2012*, *LNCS* 7237, pages 483–501. Springer, Heidelberg, April 2012.

[AJL+19] P. Ananth, A. Jain, H. Lin, C. Matt, and A. Sahai. Indistinguishability obfuscation without multilinear maps: New paradigms via low degree weak pseudorandomness and security amplification. Cryptology ePrint Archive, Report 2019/643, 2019. https://eprint.iacr.org/2019/643.

[Ajt96] M. Ajtai. Generating hard instances of lattice problems (extended abstract). In *28th ACM STOC*, pages 99–108. ACM Press, May 1996.

[AP09] J. Alwen and C. Peikert. Generating Shorter Bases for Hard Random Lattices. In *26th International Symposium on Theoretical Aspects of Computer Science STACS 2009*, Proceedings of the 26th Annual Symposium on the Theoretical Aspects of Computer Science, pages 75–86, Freiburg, Germany, February 2009. IBFI Schloss Dagstuhl.

[AP20] S. Agrawal and A. Pellet-Mary. Indistinguishability obfuscation without maps: Attacks and fixes for noisy linear FE. In *EUROCRYPT 2020, Part I*, LNCS, pages 110–140. Springer, Heidelberg, May 2020.

[BCP14] E. Boyle, K.-M. Chung, and R. Pass. On extractability obfuscation. In *TCC*, pages 52–73, 2014.

[BDGM19] Z. Brakerski, N. Döttling, S. Garg, and G. Malavolta. Leveraging linear decryption: Rate-1 fully-homomorphic encryption and time-lock puzzles. In *TCC 2019, Part II*, LNCS, pages 407–437. Springer, Heidelberg, March 2019.

[BDGM20a] Z. Brakerski, N. Döttling, S. Garg, and G. Malavolta. Candidate iO from homomorphic encryption schemes. In *EUROCRYPT 2020, Part I*, LNCS, pages 79–109. Springer, Heidelberg, May 2020.

[BDGM20b] Z. Brakerski, N. Döttling, S. Garg, and G. Malavolta. Factoring and pairings are not necessary for io: Circular-secure lwe suffices. Cryptology ePrint Archive, Report 2020/1024, 2020. https://eprint.iacr.org/2020/1024.

[BGI+01] B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S. P. Vadhan, and K. Yang. On the (im)possibility of obfuscating programs. In *CRYPTO 2001*, *LNCS* 2139, pages 1–18. Springer, Heidelberg, August 2001.

[BGL+15] N. Bitansky, S. Garg, H. Lin, R. Pass, and S. Telang. Succinct randomized encodings and their applications. *IACR Cryptology ePrint Archive*, 2015:356, 2015.

[BHW15] A. Bishop, S. Hohenberger, and B. Waters. New circular security counterexamples from decision linear and learning with errors. In *ASIACRYPT 2015, Part II*, *LNCS* 9453, pages 776–800. Springer, Heidelberg, November / December 2015.

[BIJ+20] J. Bartusek, Y. Ishai, A. Jain, F. Ma, A. Sahai, and M. Zhandry. Affine determinant programs: A framework for obfuscation and witness encryption. In *ITCS 2020*, pages 82:1–82:39. LIPIcs, January 2020.

[BP15] N. Bitansky and O. Paneth. ZAPs and non-interactive witness indistinguishability from indistinguishability obfuscation. In *TCC 2015, Part II*, *LNCS* 9015, pages 401–427. Springer, Heidelberg, March 2015.

[BPR15] N. Bitansky, O. Paneth, and A. Rosen. On the cryptographic hardness of finding a Nash equilibrium. In *56th FOCS*, pages 1480–1498. IEEE Computer Society Press, October 2015.

[BPW16] N. Bitansky, O. Paneth, and D. Wichs. Perfect structure on the edge of chaos - trapdoor permutations from indistinguishability obfuscation. In *TCC 2016-A, Part I*, *LNCS* 9562, pages 474–502. Springer, Heidelberg, January 2016.

[BR93] M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *ACM CCS 93*, pages 62–73. ACM Press, November 1993.

[BRS02] J. Black, P. Rogaway, and T. Shrimpton. Encryption-scheme security in the presence of key-dependent messages. Cryptology ePrint Archive, Report 2002/100, 2002. http://eprint.iacr.org/2002/100.

[BV15] N. Bitansky and V. Vaikuntanathan. Indistinguishability obfuscation from functional encryption. In *56th FOCS*, pages 171–190. IEEE Computer Society Press, October 2015.

[BZ14] D. Boneh and M. Zhandry. Multiparty key exchange, efficient traitor tracing, and more from indistinguishability obfuscation. In *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part I*, pages 480–499, 2014.

[CGH98] R. Canetti, O. Goldreich, and S. Halevi. The random oracle methodology, revisited (preliminary version). In *30th ACM STOC*, pages 209–218. ACM Press, May 1998.

[CGH12] D. Cash, M. Green, and S. Hohenberger. New definitions and separations for circular security. In *PKC 2012*, *LNCS* 7293, pages 540–557. Springer, Heidelberg, May 2012.

[CHJV14] R. Canetti, J. Holmgren, A. Jain, and V. Vaikuntanathan. Indistinguishability obfuscation of iterated circuits and RAM programs. Cryptology ePrint Archive, Report 2014/769, 2014. http://eprint.iacr.org/2014/769.

[CHL+15] J. H. Cheon, K. Han, C. Lee, H. Ryu, and D. Stehlé. Cryptanalysis of the multilinear map over the integers. In *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I*, pages 3–12, 2015.

[CKP15] R. Canetti, Y. T. Kalai, and O. Paneth. On obfuscation with random oracles. In *Theory of Cryptography - 12th Theory of Cryptography Conference, TCC 2015, Warsaw, Poland, March 23-25, 2015, Proceedings, Part II*, pages 456–467, 2015.

[CL01] J. Camenisch and A. Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In *EUROCRYPT 2001*, *LNCS* 2045, pages 93–118. Springer, Heidelberg, May 2001.

[CLP15] K.-M. Chung, H. Lin, and R. Pass. Constant-round concurrent zero-knowledge from indistinguishability obfuscation. In *CRYPTO 2015, Part I*, *LNCS* 9215, pages 287–307. Springer, Heidelberg, August 2015.

[CLT13] J.-S. Coron, T. Lepoint, and M. Tibouchi. Practical multilinear maps over the integers. In *CRYPTO 2013, Part I*, *LNCS* 8042, pages 476–493. Springer, Heidelberg, August 2013.

[CLT15] J.-S. Coron, T. Lepoint, and M. Tibouchi. New multilinear maps over the integers. In *CRYPTO 2015, Part I*, *LNCS* 9215, pages 267–286. Springer, Heidelberg, August 2015.

[DJ01] I. Damgård and M. Jurik. A generalisation, a simplification and some applications of Paillier's probabilistic public-key system. In *PKC 2001*, *LNCS* 1992, pages 119–136. Springer, Heidelberg, February 2001.

[Gen09] C. Gentry. Fully homomorphic encryption using ideal lattices. In *41st ACM STOC*, pages 169–178. ACM Press, May / June 2009.

[GGH13a] S. Garg, C. Gentry, and S. Halevi. Candidate multilinear maps from ideal lattices. In *EUROCRYPT 2013*, *LNCS* 7881, pages 1–17. Springer, Heidelberg, May 2013.

[GGH+13b] S. Garg, C. Gentry, S. Halevi, M. Raykova, A. Sahai, and B. Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *54th FOCS*, pages 40–49. IEEE Computer Society Press, October 2013.

[GGH15] C. Gentry, S. Gorbunov, and S. Halevi. Graph-induced multilinear maps from lattices. In *TCC 2015, Part II*, *LNCS* 9015, pages 498–527. Springer, Heidelberg, March 2015.

[GGHR14] S. Garg, C. Gentry, S. Halevi, and M. Raykova. Two-round secure MPC from indistinguishability obfuscation. In *Theory of Cryptography - 11th Theory of Cryptography Conference, TCC 2014, San Diego, CA, USA, February 24-26, 2014. Proceedings*, pages 74–94, 2014.

[GH10] M. Green and S. Hohenberger. Cpa and cca-secure encryption systems that are not 2-circular secure, 2010. matthewdgreen@gmail.com 14686 received 16 Mar 2010, last revised 18 Mar 2010.

[GJK18] C. Gentry, C. S. Jutla, and D. Kane. Obfuscation using tensor products. In *Electronic Colloquium on Computational Complexity (ECCC)*, page 149, 2018.

[GJLS20] R. Gay, A. Jain, H. Lin, and A. Sahai. Indistinguishability obfuscation from simple-to-state hard problems: New assumptions, new techniques, and simplification. Technical report, Cryptology ePrint Archive, Report 2020/764, 2020. https://eprint.iacr.org/2020/764, 2020.

[GK05] S. Goldwasser and Y. T. Kalai. On the impossibility of obfuscation with auxiliary input. In *46th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2005), 23-25 October 2005, Pittsburgh, PA, USA, Proceedings*, pages 553–562, 2005.

[GKW17] R. Goyal, V. Koppula, and B. Waters. Separating semantic and circular security for symmetric-key bit encryption from the learning with errors assumption. In *EUROCRYPT 2017, Part II*, *LNCS* 10211, pages 528–557. Springer, Heidelberg, April / May 2017.

[GLSW14] C. Gentry, A. Lewko, A. Sahai, and B. Waters. Indistinguishability obfuscation from the multilinear subgroup elimination assumption. Cryptology ePrint Archive, Report 2014/309, 2014.

[GM84] S. Goldwasser and S. Micali. Probabilistic encryption. *J. Comput. Syst. Sci.*, 28(2):270–299, 1984.

[GPV08] C. Gentry, C. Peikert, and V. Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *40th ACM STOC*, pages 197–206. ACM Press, May 2008.

[GSW13] C. Gentry, A. Sahai, and B. Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *CRYPTO 2013, Part I*, *LNCS* 8042, pages 75–92. Springer, Heidelberg, August 2013.

[ILL89] R. Impagliazzo, L. A. Levin, and M. Luby. Pseudo-random generation from one-way functions (extended abstracts). In *21st ACM STOC*, pages 12–24. ACM Press, May 1989.

[JLMS19] A. Jain, H. Lin, C. Matt, and A. Sahai. How to leverage hardness of constant-degree expanding polynomials overa $\mathbb{R}$ to build $i\mathcal{O}$. In *EUROCRYPT 2019, Part I*, *LNCS* 11476, pages 251–281. Springer, Heidelberg, May 2019.

[JLS20] A. Jain, H. Lin, and A. Sahai. Indistinguishability obfuscation from well-founded assumptions. Cryptology ePrint Archive, Report 2020/1003, 2020. https://eprint.iacr.org/2020/1003.

[JS18] A. Jain and A. Sahai. How to leverage hardness of constant-degree expanding polynomials over $\mathbb{R}$ to build iO. Cryptology ePrint Archive, Report 2018/973, 2018. https://eprint.iacr.org/2018/973.

[KLW15] V. Koppula, A. B. Lewko, and B. Waters. Indistinguishability obfuscation for turing machines with unbounded memory. In *47th ACM STOC*, pages 419–428. ACM Press, June 2015.

[KMN+14] I. Komargodski, T. Moran, M. Naor, R. Pass, A. Rosen, and E. Yogev. One-way functions and (im)perfect obfuscation. In *55th FOCS*, pages 374–383. IEEE Computer Society Press, October 2014.

[KNY14] I. Komargodski, M. Naor, and E. Yogev. Secret-sharing for NP. In *ASIACRYPT 2014, Part II*, *LNCS* 8874, pages 254–273. Springer, Heidelberg, December 2014.

[KRW15] V. Koppula, K. Ramchen, and B. Waters. Separations in circular security for arbitrary length key cycles. In *TCC 2015, Part II*, *LNCS* 9015, pages 378–400. Springer, Heidelberg, March 2015.

[KW16] V. Koppula and B. Waters. Circular security separations for arbitrary length cycles from LWE. In *CRYPTO 2016, Part II*, *LNCS* 9815, pages 681–700. Springer, Heidelberg, August 2016.

[Lin16] H. Lin. Indistinguishability obfuscation from constant-degree graded encoding schemes. In *EUROCRYPT 2016, Part I*, *LNCS* 9665, pages 28–57. Springer, Heidelberg, May 2016.

[Lin17] H. Lin. Indistinguishability obfuscation from SXDH on 5-linear maps and locality-5 PRGs. In *CRYPTO 2017, Part I*, *LNCS* 10401, pages 599–629. Springer, Heidelberg, August 2017.

[LPST16] H. Lin, R. Pass, K. Seth, and S. Telang. Indistinguishability obfuscation with non-trivial efficiency. In *PKC 2016, Part II*, *LNCS* 9615, pages 447–462. Springer, Heidelberg, March 2016.

[LT17] H. Lin and S. Tessaro. Indistinguishability obfuscation from trilinear maps and block-wise local PRGs. In *CRYPTO 2017, Part I*, *LNCS* 10401, pages 630–660. Springer, Heidelberg, August 2017.

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

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

[Mic01] D. Micciancio. Improving lattice based cryptosystems using the hermite normal form. In *Cryptography and Lattices Conference — CaLC 2001*, *Lecture Notes in Computer Science* 2146, pages 126–145, Providence, Rhode Island, 29–30March 2001. Springer-Verlag.

[Mic19] D. Micciancio. From linear functions to fully homomorphic encryption. https://bacrypto.github.io/presentations/2018.11.30-micciancio-fhe.pdf. Technical report, 2019.

[MMN15] M. Mahmoody, A. Mohammed, and S. Nematihaji. More on impossibility of virtual black-box obfuscation in idealized models. *IACR Cryptology ePrint Archive*, 2015:632, 2015.

[MO14] A. Marcedone and C. Orlandi. Obfuscation ⇒ (IND-CPA security ⇏ circular security). In *SCN 14*, *LNCS* 8642, pages 77–90. Springer, Heidelberg, September 2014.

[MP12] D. Micciancio and C. Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In *EUROCRYPT 2012*, *LNCS* 7237, pages 700–718. Springer, Heidelberg, April 2012.

[MRH04] U. M. Maurer, R. Renner, and C. Holenstein. Indifferentiability, impossibility results on reductions, and applications to the random oracle methodology. In *TCC 2004*, *LNCS* 2951, pages 21–39. Springer, Heidelberg, February 2004.

[Pai99] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *EUROCRYPT'99*, *LNCS* 1592, pages 223–238. Springer, Heidelberg, May 1999.

[PRS17] C. Peikert, O. Regev, and N. Stephens-Davidowitz. Pseudorandomness of ring-LWE for any ring and modulus. In *49th ACM STOC*, pages 461–473. ACM Press, June 2017.

[Ps16] R. Pass and a. shelat. Impossibility of VBB obfuscation with ideal constant-degree graded encodings. In *TCC 2016-A, Part I*, *LNCS* 9562, pages 3–17. Springer, Heidelberg, January 2016.

[PST14] R. Pass, K. Seth, and S. Telang. Indistinguishability obfuscation from semantically-secure multilinear encodings. In *CRYPTO 2014, Part I*, *LNCS* 8616, pages 500–517. Springer, Heidelberg, August 2014.

[PVW08] C. Peikert, V. Vaikuntanathan, and B. Waters. A framework for efficient and composable oblivious transfer. In *CRYPTO 2008*, *LNCS* 5157, pages 554–571. Springer, Heidelberg, August 2008.

[Reg05] O. Regev. On lattices, learning with errors, random linear codes, and cryptography. In *37th ACM STOC*, pages 84–93. ACM Press, May 2005.

[Rot13] R. Rothblum. On the circular security of bit-encryption. In *TCC 2013*, *LNCS* 7785, pages 579–598. Springer, Heidelberg, March 2013.

[SW14] A. Sahai and B. Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In *46th ACM STOC*, pages 475–484. ACM Press, May / June 2014.

[WW20] H. Wee and D. Wichs. Candidate obfuscation via oblivious lwe sampling. Cryptology ePrint Archive, Report 2020/1042, 2020. https://eprint.iacr.org/2020/1042.

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

A XiO from the 2-circular SRL Security of the GSW FHE and the DJ LHE

A.1 Hintable LHE from DCR

We consider the Damgård Jurik (DJ) Linearly Homomorphic Encryption scheme from [DJ01], which generalizes Paillier's encryption scheme [Pai99] to larger message spaces, whose security relies on the Decisional Composite Residuosity (DCR) assumption. This scheme is not needed for our main result,

but serves as a good warm-up for understanding the notion of a hintable packed LHE (and leads to our simplest construction of an $Xi\mathcal{O}$). As mentionned in the introduction, BDGM already showed that the DJ scheme is a hintable LHE; since our notion of a hintable LHE is somewhat different, for completeness, we here include the standard proof. We start by recalling the DCR assumption under which the DJ scheme is proven secure.

Definition A.1 (Decisional Composite Residuosity (DCR) assumption [Pai99])**.** *We say that security of the DCR assumption holds if there exists a PPT algorithm* $\mathsf{RSAsample}$ *that on input a security parameter* λ*, outputs a pair* $(N, \phi(N))$ *where* N *is a* 2λ*-bits integer,* ϕ *denotes Euler's totient function; such that* $\gcd(\phi(N), N) = 1$ *and such that for all polynomial* $\zeta(\cdot)$*, the following ensembles are computationally indistinguishable:*

$$\{\mathcal{D}^0_\lambda\}_{\lambda\in\mathbb{N}} = \left\{(N,\phi(N)) \leftarrow \mathsf{RSAsample}(1^\lambda); r \leftarrow_{\mathsf{R}} \mathbb{Z}_M : r^{N^{\zeta}(\lambda)} \in \mathbb{Z}_{N^{\zeta(\lambda)+1}}\right\}_{\lambda\in\mathbb{N}}$$
$$\{\mathcal{D}^0_\lambda\}_{\lambda\in\mathbb{N}} = \left\{(N,\phi(N)) \leftarrow \mathsf{RSAsample}(1^\lambda); u \leftarrow_{\mathsf{R}} \mathbb{Z}_{N^{\zeta(\lambda)+1}} : u \in \mathbb{Z}_{N^{\zeta(\lambda)+1}}\right\}_{\lambda\in\mathbb{N}}$$

We say the DCR assumption holds if 1*-security of the LWE assumption holds, and that the subexponential DCR assumption holds if there exists* $\varepsilon > 0$ *such that* 2^{λ^ε}*-security of the DCR assumptions holds.*

As explained in [DJ01] in further details, the algorithm $\mathsf{RSAsample}(1^\lambda)$ samples two safe primes p, q of λ bits each, and compute the RSA modulus $N = pq$.

We will show how the DJ encryption scheme satisfies our notion of a hintable packed LHE (actually even without any packing):

Theorem A.2. *Assume (subexponential) security of the DCR assumption. Then, for every polynomial* ℓ_1*, there exists a (subexponentially) secure* $(\ell_1, \ell_2, h, \alpha)$*-hintable packed LHE, where* $h(\lambda) = 2\lambda$*,* $\ell_2(\lambda) = 1$*, and* $\alpha(\lambda) = 0$*.*

A.1.1 The DJ scheme

Given a polynomial $\ell_1(\cdot)$, we recall the LHE from [DJ01] when operating on plaintext of size ℓ_1 (recall that $\ell_2 = 1$ so there is no packing). That is, the scheme is parameterized by a polynomial ℓ_1, and we call it DJ_{ℓ_1}. For this scheme, the hint is simply the randomness used to encrypt a message, and noisy and normal encryptions behave in the same way:

• $\underline{\mathsf{CRSgen}(1^\lambda)}$:
It simply outputs $\mathsf{crs} = 1^\lambda$, i.e. there is no proper crs for that scheme.

• $\underline{\mathsf{Gen}(\mathsf{crs})}$:
Given $\mathsf{crs} = 1^\lambda$, it uses the sampling algorithm from Definition A.1, $(M, \phi(M)) \leftarrow \mathsf{RSAsample}(1^\lambda)$, where $M \in \mathbb{N}$ is a 2λ-bit modulus, ϕ denotes Euler's totient function, and we have $\gcd(\phi(M), M) = 1$. Then it chooses a polynomial $\zeta(\cdot)$ such that $2^{\ell_1(\lambda)+2\lambda} > N \geq 2^{\ell_1(\lambda)}$, where $N = M^{\zeta(\lambda)}$. For simplicity of the notations, we write $\zeta = \zeta(\lambda)$. It sets $\mathsf{pk} = (N, \zeta)$, $\mathsf{sk} = \mathsf{td} = \phi(M)$ and outputs $(\mathsf{pk}, \mathsf{sk}, \mathsf{td})$. The plaintext space is $\mathbb{Z}_N = \mathbb{Z}_{M^\zeta}$. The randomness space for Enc is $\mathbb{Z}^*_M$, the ciphertext space is $\mathbb{Z}^*_{M^\zeta}$, and the function space is $\mathbb{Z}_N$, that is $t = N$.

• $\underline{\mathsf{Enc}_{\mathsf{pk}}(\mathbf{x})}$:
Given the public pk, a vector $\mathbf{x} \in \mathbb{Z}^\nu_N$, for all $i \in [\nu]$, it samples $r_i \leftarrow_{\mathsf{R}} \mathbb{Z}^*_M$ and compute $\mathsf{ct}_i = r_i^{M^\zeta} \cdot (1+M)^{x_i} \in \mathbb{Z}^*_{M^{\zeta+1}}$. It outputs the ciphertext $\mathsf{ct} = (\mathsf{ct}_1, \ldots, \mathsf{ct}_\nu)$.

• $\underline{\mathsf{Enc}^{\star}_{\mathsf{pk}}(m)}$:
Given the public pk, a message $m \in \mathbb{Z}_N$, it samples $r \leftarrow_{\mathsf{R}} \mathbb{Z}^*_M$ and outputs the noisy ciphertext $\mathsf{ct}^{\star} = r^{M^{\zeta}} \cdot (1+M)^m \in \mathbb{Z}^*_{M^{\zeta+1}}$.

• $\underline{\mathsf{Eval}(\mathsf{pk}, \mathsf{ct}, \mathsf{ct}^{\star}, \mathbf{y})}$:
Given as input the public key pk, ciphertext $\mathsf{ct} \in \mathbb{Z}^{*\nu}_{M^{\zeta+1}}$, noisy ciphertext $\mathsf{ct}^{\star} \in \mathbb{Z}^*_{M^{\zeta+1}}$, and a vector $\mathbf{y} \in [-1,1]^{\nu}$, it outputs the evaluated ciphertext $\mathsf{ct}^{\star} \cdot \prod_{i\in[\nu]} \mathsf{ct}_i^{y_i} \in \mathbb{Z}^*_{M^{\zeta+1}}$, where $\cdot$ denotes the integer multiplication in $\mathbb{Z}^*_{M^{\zeta+1}}$.

• $\underline{\mathsf{SecHint}(\mathsf{td}, \mathsf{ct}^{\star})}$:
Given the secret key td and a noisy ciphertext $\mathsf{ct}^{\star} \in \mathbb{Z}^*_{M^{\zeta+1}}$, it computes $d = \mathsf{ct} \mod M$. Since $\gcd(M^{\zeta}, \phi(M)) = 1$, it can compute $M^{-\zeta} \in \mathbb{Z}$ such that $M^{\zeta} \cdot M^{-\zeta} = 1 \mod \phi(M)$. It outputs the hint $d^{M^{-\zeta}} \in \mathbb{Z}^*_M$.

• $\underline{\mathsf{PubHint}(\mathsf{pk}, r)}$:
Given the public key pk and some randomness $r \in Z^*_M$, it outputs the hint $\rho = r$.

• $\underline{\mathsf{Rec}(\mathsf{pk}, \mathsf{ct}^{\star}, \rho)}$:
Given the public key pk, a noisy ciphertext $\mathsf{ct}^{\star}$, and a hint $\rho \in \mathbb{Z}^*_M$, it computes $d = \mathsf{ct} \cdot r^{-M^{\zeta}} \in \mathbb{Z}^*_{M^{\zeta+1}}$, where $\rho^{-M^{\zeta}}$ is the inverse of $\rho^{M^{\zeta}}$ in $\mathbb{Z}^*_{M^{\zeta+1}}$. Then, it applies Paillier's decryption recursively to obtain $x \in \mathbb{Z}_{M^{\zeta}}$. It outputs $x \in \mathbb{Z}_N$.

• $\underline{\mathsf{Dec}_{\mathsf{sk}}(\mathsf{ct}^{\star})}$:
Given the secret key sk, a noisy ciphertext $\mathsf{ct}^{\star}$, it runs $\mathsf{Ext}(\mathsf{sk}, \mathsf{ct}^{\star})$ to recover the randomness $r \in \mathbb{Z}^*_M$, then outputs $\mathsf{Rec}(\mathsf{pk}, \mathsf{ct}^{\star}, r)$.

The proof of Theorem A.2 follows from the propositions and theorem below (which demonstrate that DJ satisfies the desired properties of a hintable packed LHE, as well as security).

Proposition 14 (Linear Homomorphism)**.** *The LHE presented above satisfies Property 4.2.*

Proof: For all $i \in [\nu]$, let ct_i be a ciphertext in the support of $\mathsf{Enc}_{\mathsf{pk}}(x_i)$, that is, of the form $\mathsf{ct}_i = (r_i)^{M^{\zeta}} \cdot (1+M)^{x_i} \in \mathbb{Z}^*_{M^{\zeta+1}}$, and let $\mathsf{ct}^{\star}$ be a ciphertext in the support of $\mathsf{Enc}^{\star}_{\mathsf{pk}}(x^{\star})$, that is, of the form $\mathsf{ct}^{\star} = r^{M^{\zeta}} \cdot (1+M)^{x^{\star}} \in \mathbb{Z}^*_{M^{\zeta+1}}$. For all $\mathbf{y} \in \{0,1\}^{\nu}$, the evaluated ciphertext $\mathsf{ct}_{\mathbf{y}}$ is of the form: $\left(r \prod_{i\in[\nu]} r_i^{y_i}\right)^{M^{\zeta}} \cdot (1+M)^{x^{\star} + \sum_{i\in[\nu]} x_i y_i} \in \mathbb{Z}^*_{M^{\zeta+1}}$, which is in the support of $\mathsf{Enc}^{\star}_{\mathsf{pk}}(x^{\star} + \sum_{i\in[\nu]} x_i y_i)$. □

Proposition 15 (0-approximate correctness of secret hints)**.** *The LHE presented above satisfies* 0*-approximate correctness, as defined in Property 4.3.*

Proof: Let $\mathsf{ct}^{\star} = r^{M^{\zeta}} \cdot (1+M)^{x^{\star}} \in \mathbb{Z}^*_{M^{\zeta+1}}$, where $x^{\star} \in \mathbb{Z}_{M^{\zeta}}$ is the message and $r \in \mathbb{Z}^*_M$ is the randomness used to produce the ciphertext. We have $\mathsf{ct}^{\star} \mod M = r^{M^{\zeta} \mod \phi(M)} \in \mathbb{Z}^*_M$. Since $\gcd(\phi(M), M^{\zeta}) = 1$, there is $M^{-\zeta} \in \mathbb{Z}$ such that $M^{\zeta} \cdot M^{-\zeta} = 1 \mod \phi(M)$. Thus, the algorithm $\mathsf{SecHint}(\mathsf{td}, \mathsf{ct})$ outputs $r \in \mathbb{Z}^*_M$. That is, the hint output by $\mathsf{SecHint}$ is the randomness used to produce $\mathsf{ct}^{\star}$.

The algorithm $\mathsf{Rec}(\mathsf{pk}, \mathsf{ct}^{\star}, r)$ computes $d = \mathsf{ct}^{\star} \cdot r^{-M^{\zeta}} = (1+M)^{x^{\star}} \in \mathbb{Z}^*_{M^{\zeta+1}}$ where $r^{-M^{\zeta}}$ is the inverse of $r^{M^{\zeta}}$ in $\mathbb{Z}^*_{M^{\zeta+1}}$. Then it applies Paillier's decryption recursively to obtain $x^{\star} \in \mathbb{Z}_{M^{\zeta}}$. It outputs $x^{\star}$. □

Proposition 16 (Equivalence between public hints and secret hints)**.** *The LHE presented above satisfies Property 4.4.*

Proof: As seen in the proof of Proposition 15, the hint recovered by $\mathsf{SecHint}$ given the trapdoor td and a ciphertext $\mathsf{ct}^\star$ is the randomness r used by $\mathsf{Enc}^\star$ to produce $\mathsf{ct}^\star$, which is also what $\mathsf{PubHint}(\mathsf{pk}, r)$ outputs. □

Proposition 17 (0-approximate correctness)**.** *The LHE presented above satisfies* 0*-approximate correctness, as defined in Property 4.1.*

Proof: This directly follows from the 0-approximate correctness of the secret hints (Property 4.3), since decryption first computes the hint using the secret key, then recovers the message using the hint. □

Proposition 18 (h-succinctness of hints)**.** *The LHE presented above satisfies $h(\lambda) = 2\lambda$-succinctness.*

Proof: For all $\lambda \in \mathbb{N}$, for all pairs $(\mathsf{pk}, \mathsf{sk})$ in the support of $\mathsf{Gen}(1^\lambda)$ that define the message space $\mathbb{Z}_N$, for all $x^\star \in \mathbb{Z}_N$, all ciphertext $\mathsf{ct}^\star$ in the support of $\mathsf{Enc}^\star_{\mathsf{pk}}(x^\star)$, we have: all hints ρ in the support of $\mathsf{SecHint}(\mathsf{sk}, \mathsf{ct}^\star)$ are in $\mathbb{Z}^*_M$, where M is an RSA modulus of size at most 2λ bits. □

Proposition 19 (Weak circuit privacy)**.** *The LHE presented above satisfies Property 4.6 (weak circuit privacy).*

Proof: For any message $\mathbf{x} = (x_1, \ldots, x_\nu) \in \mathbb{Z}_N^\nu$, $x^\star \in \mathbb{Z}_N$, $\mathbf{y} \in \{0,1\}^\nu$, we aim at proving that following distributions are identical:

$$\mathcal{D}_0 : \left\{ \begin{array}{l} \forall i \in [\nu], r_i \leftarrow_{\mathsf{R}} \mathbb{Z}^*_M, \mathsf{ct}_i = r_i^{M^\zeta} \cdot (1+M)^{x_i}, r \leftarrow_{\mathsf{R}} \mathbb{Z}^*_M, \mathsf{ct}^\star = r^{M^\zeta} \cdot (1+M)^{x^\star} \\ \mathsf{ct}_{\mathbf{y}} = (r \prod_i r_i^{y_i})^{M^\zeta} \cdot (1+M)^{x^\star + \mathbf{x}^\top \mathbf{y}} : ((\mathsf{ct}_i)_i, \mathsf{ct}^\star, \mathsf{ct}_{\mathbf{y}}) \end{array} \right\}$$

$$\mathcal{D}_1 : \left\{ \begin{array}{l} \forall i \in [\nu], r_i \leftarrow_{\mathsf{R}} \mathbb{Z}^*_M, \mathsf{ct}_i = r_i^{M^\zeta} \cdot (1+M)^{x_i}, r \leftarrow_{\mathsf{R}} \mathbb{Z}^*_M, \mathsf{ct}^\star = (r/\prod_i r_i^{y_i})^{M^\zeta} \cdot (1+M)^{x^\star} \\ \mathsf{ct}_{\mathbf{y}} = r^{M^\zeta} \cdot (1+M)^{x^\star + \mathbf{x}^\top \mathbf{y}} : ((\mathsf{ct}_i)_i, \mathsf{ct}^\star, \mathsf{ct}_{\mathbf{y}}) \end{array} \right\}.$$

This relies on the fact that for all $i \in [\nu]$, all $r_i \in \mathbb{Z}^*_M$, all $y_i \in \mathbb{Z}_{M^\zeta}$, the following distributions are identical: $\mathcal{D}'_0 = \{r \leftarrow_{\mathsf{R}} \mathbb{Z}^*_M : ((r_i)_i, r, r \cdot \prod_i r_i^{y_i})\}$ and $\mathcal{D}'_1 = \{r \leftarrow_{\mathsf{R}} \mathbb{Z}^*_M : ((r_i)_i, r/\left(\prod_i r_i^{y_i}\right), r)\}$. The distribution $\mathcal{D}'_0$ corresponds to $\mathcal{D}_0$, whereas $\mathcal{D}'_1$ corresponds to $\mathcal{D}_1$. □

Proposition 20 (Density of the noisy ciphertexts)**.** *The LHE presented above satisfies Property 4.7 (density of the noisy ciphertexts).*

Proof: One can sample a uniform random value $u \leftarrow_{\mathsf{R}} \mathbb{Z}_{M^{\zeta+1}}$ from $\lceil \log(M^{\zeta+1}) \rceil$ random bits. The random value $u \in \mathbb{Z}_{M^{\zeta+1}}$ can be written $u = r^{M^\zeta} \cdot (1+M)^x$ where $x \in \mathbb{Z}_{M^\zeta}$ and $r \in \mathbb{Z}^*_M$ with probability $1 - \frac{\phi(N)}{N} > 1 - \frac{3}{2^\lambda}$ over the choice of $u \leftarrow_{\mathsf{R}} \mathbb{Z}_{M^{\zeta+1}}$. □

Theorem A.3 (Security [DJ01])**.** *Assuming the (subexponential) DCR assumption, the DJ scheme is (subexponentially) secure.*

A.2 Instantiation with DJ LHE

Now we instantiate the modular XiO construction described in Section 5.2 with the DJ LHE presented in Section A.1 that is parameterized by a polynomial ℓ_1; we denote it by DJ_{ℓ_1}. For all polynomials δ, we denote by GSW_δ the GSW FHE scheme for depth δ circuits.

Before stating our theorems, we need to define the analogue of the $\mathsf{1CIRC}$ assumption (given in 2.10) in the context of 2-circular security.

Definition A.4 (2CIRC assumption)**.** *We say that the* (subexponential) $\mathsf{2CIRC}^{\mathcal{O}}$ assumption holds w.r.t $\mathcal{PKE}$ and $\overline{\mathcal{PKE}}$ *if the following holds: if $\mathcal{PKE}$ is (subexponentially) $\mathcal{O}$-leakage resilient secure and $\overline{\mathcal{PKE}}$ is (subexponentially) secure, then (subexponential) $\mathcal{O}$-leakage resilient 2-circular security holds w.r.t $\mathcal{PKE}$ and $\overline{\mathcal{PKE}}$.*

By combining Theorem A.3 (i.e. security of DJ_{ℓ_1} for all polynomials ℓ_1 under DCR) and Theorem 3.5 (i.e. SRL-security of the GSW_δ for all polynomials δ under LWE), we get:

Lemma A.1. *Assume the (subexponential) DCR and the (subexponential) LWE assumptions hold. Assume further that for all polynomials δ and ℓ_1, the $\mathsf{2CIRC}^{\mathcal{O}_{\mathsf{SRL}}}$ assumption holds w.r.t. GSW_δ and DJ_{ℓ_1}. Then, for all polynomials δ and ℓ_1, (subexponential) 2-circular SRL security holds w.r.t. GSW_δ and DJ_{ℓ_1}.*

By combining Lemma A.1 above with Theorem A.2, which states that for all polynomials ℓ_1, DJ_{ℓ_1} is an $(\ell_1, \ell_2, h, \alpha)$-hintable packed LHE with $\ell_2(\lambda) = 1$, $h(\lambda) = 2\lambda$ and $\alpha(\lambda) = 0$, together with Theorem 5.3, we get:

Theorem A.5. *Assume the (subexponential) DCR and (subexponential) LWE assumptions hold. Assume further that for all polynomials δ, ℓ_1, the $\mathsf{2CIRC}^{\mathcal{O}_{\mathsf{SRL}}}$ assumption holds w.r.t. GSW_δ and DJ_{ℓ_1}. Then (subexponentially-secure) $Xi\mathcal{O}$ for $\mathsf{P}^{\log}/\mathsf{poly}$ exists.*

Finally, combining Theorem A.5 with Theorem 2.4 (i.e., $i\mathcal{O}$ from $Xi\mathcal{O}$ and LWE) yields one of our two main theorem:

Theorem A.6. *Assume the subexponential DCR and subexponential LWE assumptions hold. Assume further that that for all polynomials δ and ℓ_1 the $\mathsf{2CIRC}^{\mathcal{O}_{\mathsf{SRL}}}$ holds w.r.t. GSW_δ and DJ_{ℓ_1}. Then subexponentially-secure $i\mathcal{O}$ for P/poly exists.*

B Concrete Assumption

In Theorem 5.6, we show that subexponential LWE assumption and the subexponential $\mathsf{1CIRC}^{\mathcal{O}_{\mathsf{SRL}}}$ assumption w.r.t. GSW_δ for all polynomials δ implies subexponentially-secure $i\mathcal{O}$ for P/poly.

As explained in Section 3.2.3, it actually suffices to assume that for all polynomials δ, semantic security of GSW_δ implies the following variant of SRL circular security (that is weaker than that of definition Definition 3.3) of GSW_δ.

Definition B.1 (One-shot SRL circular security)**.** *We say that an FHE scheme $\mathcal{FHE} = (\mathsf{CRSgen}, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Enc}^\star, \mathsf{Eval}', \mathsf{Eval}_{\mathsf{rand}}, \mathsf{Dec})$ that satisfies batch correctness (as per Definition 3.1) and randomness homomorphism (as per Definition 3.2) is one-shot SRL circular secure if for all stateful nuPPT adversaries $\mathcal{A}$, there exists some negligible function $\mu(\cdot)$ such that for all $\lambda \in \mathbb{N}$, $\Pr[\mathsf{Exp}^{\mathcal{FHE}}_{\lambda,\mathcal{A}} = 1] \leq 1/2 + \mu(\lambda)$, where the experiment $\mathsf{Exp}^{\mathcal{FHE}}_{\lambda,\mathcal{A}}$ is defined as follows:*

$$\mathsf{Exp}^{\mathcal{FHE}}_{\lambda,\mathcal{A}} = \left\{ \begin{array}{l} (\mathbf{m}^0, \mathbf{m}^1, q) \leftarrow \mathcal{A}(1^\lambda), \mathsf{crs} \leftarrow \mathsf{CRSgen}(1^\lambda) \\ (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(\mathsf{crs}), b \leftarrow \{0,1\} \\ \mathbf{r} \leftarrow_R \mathcal{R}^{|\mathsf{sk}\|\mathbf{m}^b|}, \mathsf{ct} = \mathsf{Enc}_{\mathsf{pk}}(\mathsf{sk}\|\mathbf{m}^b; \mathbf{r}) \\ \forall i \in [q(\lambda)] : r^\star_i \leftarrow_R \mathcal{R}^\star, \mathsf{ct}_i = \mathsf{Enc}^\star_{\mathsf{pk}}(\mathbf{0}; \mathbf{r}^\star_i) \\ (f_i, \alpha_i)_{i \in [q(\lambda)]} \leftarrow \mathcal{A}(\mathsf{pk}, \mathsf{ct}) \\ \forall i \in [q(\lambda)], \mathbf{r}_{f_i} = \mathsf{Eval}_{\mathsf{rand}}(\mathsf{pk}, f_i, \mathbf{r}, \mathsf{sk}\|\mathbf{m}^b), \mathsf{leak}_i = \mathbf{r}^\star_i - \mathbf{r}_{f_i} \\ b' \leftarrow \mathcal{A}\big((\mathsf{leak}_i)_{i \in [q(\lambda)]}\big) \\ \textit{Return 1 if } |\mathbf{m}^0| = |\mathbf{m}^1|,\ b' = b \textit{ and for all } i \in [p(\lambda)],\ f_i(\mathsf{sk}\|\mathbf{m}^b) = \alpha_i\textit{; 0 otherwise.} \end{array} \right\}$$