Indistinguishability Obfuscation Without Multilinear Maps: New Paradigms via Low Degree Weak Pseudorandomness and Security Amplification[*]

Prabhanjan Ananth[1], Aayush Jain[2], Huijia Lin[3],
Christian Matt[4], and Amit Sahai[2]

[1] MIT
prabhanjan@csail.mit.edu
[2] UCLA
{aayushjain,sahai}@cs.ucla.edu
[3] University of Washington, Seattle
rachel@cs.washington.edu
[4] Concordium, Zurich, Switzerland
cm@concordium.com

Abstract. The existence of secure indistinguishability obfuscators ($i\mathcal{O}$) has far-reaching implications, significantly expanding the scope of problems amenable to cryptographic study. All known approaches to constructing $i\mathcal{O}$ rely on *d-linear maps*. While secure *bilinear maps* are well established in cryptographic literature, the security of candidates for $d > 2$ is poorly understood.

We propose a new approach to constructing $i\mathcal{O}$ for general circuits. Unlike all previously known realizations of $i\mathcal{O}$, we avoid the use of d-linear maps of degree $d \geq 3$.

At the heart of our approach is the assumption that a new *weak* pseudorandom object exists. We consider two related variants of these objects, which we call *perturbation resilient generator* (ΔRG) and *pseudo flawed-smudging generator* (PFG), respectively. At a high level, both objects are polynomially expanding functions whose outputs partially hide (or smudge) small noise vectors when added to them. We further require that they are computable by a family of degree-3 polynomials over $\mathbb{Z}$. We show how they can be used to construct functional encryption schemes with weak security guarantees. Finally, we use novel amplification techniques to obtain full security.

As a result, we obtain $i\mathcal{O}$ for general circuits assuming:

- Subexponentially secure LWE
- Bilinear Maps
- poly(λ)-secure 3-block-local PRGs
- ΔRGs or PFGs

---

[*] This paper is a merge of two independent works, one by Ananth, Jain, and Sahai [AJS18], and the other by Lin and Matt [LM18].

1   Introduction

Program obfuscation considers the problem of building an efficient randomized compiler that takes as input a computer program P and outputs an equivalent program $O(P)$ such that any secrets present within P are "as hard as possible" to extract from $O(P)$. This property can be formalized by the notion of indistinguishability obfuscation ($i\mathcal{O}$) [BGI+01, GR07]. Formally, $i\mathcal{O}$ requires that given any two equivalent programs P_1 and P_2 of the same size, it is not possible for a computationally bounded adversary to distinguish between the obfuscated versions of these programs. Recently, starting with the works of [GGH+13b, SW14], it has been shown that $i\mathcal{O}$ would have far-reaching applications, significantly expanding the scope of problems to which cryptography can be applied (e.g., [SW14, KLW15, GGHR14, CHN+16, GPS16, HSW14, BPR15, GGG+14, HJK+16, BFM14]).

The work of [GGH+13b] gave the first mathematical candidate $i\mathcal{O}$ construction, and since then more than a dozen candidates have been proposed and studied [GGH13a, CLT13a, GGH15a, CLT15a, Hal15, BR14a, BGK+14a, PST14a, AGIS14a], [BMSZ16, CHL+15, BWZ14, CGH+15, HJ15, BGH+15, Hal15, CLR15, MF15, MSZ16, DGG+16a]. Furthermore, more recent candidates [Lin16a, LV16, AS17, LT17] based $i\mathcal{O}$ on simple primtives and assumptions. However, all these $i\mathcal{O}$ constructions rely on multi-linear maps with degree at least 3. Unfortunately, all known candidates for degree-3 multilinear maps [GGH13a, CLT13a, GGH15a] have poorly understood security properties, and even security models [MSZ16, BGMZ18, MZ18].

Our results in a nutshell. Securely building $i\mathcal{O}$ remains a central challenge in cryptography. In this paper, we report on the works of [AJS18, LM18], in which we develop new techniques that enables building $i\mathcal{O}$ without multilinear maps of degree ≥ 3. Instead, we rely on (relatively) standard assumptions including (subexponentnaily secure) *bilinear maps*, LWE, and block-local PRGs [LT17] (a relaxation of local PRGs, a.k.a. Goldreich's PRGs [Gol00]), as well as new types of "weak" pseudo-randomness generators with certain "simple" structures — either perturbation resilient generators [AJS18] or pseudo flawed-smudging generators [LM18].

Along the way, we study the notion of Functional Encryption, which was introduced by [SW05], and formalized by [BSW11, O'N10]. We provide new general security amplification theorems for amplifying Functional Encryption with $(1/\lambda^c)$-indistinguishability-based security to Functional Encryption with standard security [AJS18], and security amplification for amplifying certain leaky forms of Functional Encryption to standard security [LM18]. We now elaborate.

Prior $i\mathcal{O}$ from multilinear maps with degree ≥ 3. The first-generation $i\mathcal{O}$ constructions [GGH+13b, BR14b, BGK+14b, PST14b, AGIS14b, GLSW14, Zim15, AB15, GMM+16a, DGG+16b] rely on polynomial-degree *multilinear maps* or *graded encodings*. An L-linear map [BS02] essentially allows to evaluate degree-L polynomials on secret encoded values, and to test whether the output of such polynomials is zero or not. While bilinear maps (i.e., $L = 2$) can be efficiently

instantiated from elliptic curves, instantiation of L-linear maps for $L \geq 3$ has remained elusive—While candidate constructions of such graded encoding schemes exist [CLT13b, LSS14, GGH15b, CLT15b], their security is poorly understood due to several known explicit attacks on certain distributions of encoded values [CHL⁺15, BWZ14, CGH⁺15, HJ15, BGH⁺15, Hal15, CLR15, MF15, MSZ16] [5].

A line of recent works [Lin16b, LV16, Lin17, AS17] aimed at finding the minimal degree of multilinear maps sufficient for constructing $i\mathcal{O}$, and has successfully reduced the required degree to $L = 3$. A key ingredient in these second-generation constructions are PRGs with *small locality*[6]. They showed that to construct $i\mathcal{O}$, it suffices to have multilinear maps with degree matching exactly the locality of the PRG [Lin16b, AS17], or even the relaxed notion of *block locality* [LT17]. These constructions essentially use degree-L multilinear maps to evaluate a PRG with (block-)locality L, and then bootstrap from there to hide arbitrary complex computation. Unfortunately, the locality of a PRG cannot be smaller than 5 [CM01, MST03], and recent attacks [LV17, BBKK18] showed that block-locality cannot be smaller than 3.[7] This raises the following natural question:

Can we build $i\mathcal{O}$ without cryptographic multilinear maps of degree ≥ 3?
Are there new types of simple and weak pseudo-randomness generators
that can help?

Our simple and weak pseudorandomness generators. We answer the above questions positively, relying on either the new notion of *perturbation-resilient generators*, ΔRG for short, proposed by [AJS18] (AJS), or *pseudo flawed-smuding generators*, PFG for short, proposed by [LM18] (LM). They are weak pseudo-randomness generators with the same simple structure, and similar intuitive security guarantees. However, their concrete security formalizations are very different, requiring different techniques of using them in $i\mathcal{O}$ constructions as done in [AJS18, LM18].

We start with explaining their shared simple structure. A ΔRG/PFG is given by a polynomially expanding function G from n input (or seed) elements to $m = n^{1+\alpha}$ output elements in $\mathbb{Z}_p$, together with a seed distribution $\mathcal{S}$ over $\mathbb{Z}_p^n$ that samples a pair $\mathbf{s} = (\mathbf{s}_1, \mathbf{s}_2)$ of public and private seeds[8]. G has the simple structure that *1)* it is a degree 3 polynomial over $\mathbb{Z}_p$ with degree 1 in the public seed $\mathbf{s}_1$ and degree 2 in the private seed $\mathbf{s}_2$, and *2)* its output distribution $G(\mathcal{S})$ is polynomially bounded. At a very high-level, these two structural properties

---

[5] Note that this does not necessarily mean that the resulting $i\mathcal{O}$ constructions are insecure; in particular, there have been efforts (e.g., [GMM⁺16b]) in constructing $i\mathcal{O}$ in more complex security models for multilinear maps (e.g. [MSZ16]) that have resisted polynomial-time attacks. There have also been several other $i\mathcal{O}$ candidates proposed which are not known to polynomial-time broken (e.g. [CVW18, BGMZ18]).

[6] A function has locality ℓ if every output element depends on at most ℓ input elements.

[7] The attacks actually leave open a very small window of expansion. Nevertheless, they have weakened our confidence on the security of PRGs with block-locality 2.

[8] n, m, p are parameterized by the security parameter λ

ensure that we can essentially compute G in the exponent of bilinear pairing groups (property 1) and extract the output in the clear via brute force discrete logarithm (property 2). An acute reader may be curious about the purpose of the public seed $\mathbf{s}_1$. In short, it is a relaxation to requiring G having total degree 2, and as we shall see later, is crucial for the security of the instantiation of G.

Intuitively, the security of ΔRG/PFG guarantees that its output when added to a small noise vector, producing $G(\mathbf{s}) + \mathbf{e}$, *weakly* "smudge" or "hide" $\mathbf{e}$. In the literature, noise smudging (or noise flooding) is a commonly used technique for hiding small noises in LWE samples, which is also our purpose. However, to completely hide the noise vector $\mathbf{e}$, the smudging noises must be super-polynomially large. This stands in contrast with the fact that $G(\mathbf{s})$ is polynomially bounded. To circumvent this, ΔRG and PFG formalizes different weakly hiding requirements:

- ΔRG guarantees that the distributions $\Delta RG(\mathbf{s})$ and $(\Delta RG(\mathbf{s}) + \mathbf{e})$ are *somewhat* hard to distinguish as long as the perturbation $\mathbf{e}$ is relatively small. More specifically, it suffices if efficient adversaries fail to distinguish these two distributions with at least some fixed $1/\text{poly}(\lambda)$ probability. Thus, a candidate ΔRG would be secure, for instance, if an adversary could distinguish between $\Delta RG(\mathbf{s})$ and $(\Delta RG(\mathbf{s}) + \mathbf{e})$ with probability 99%, but no adversary could distinguish with probability over 99.5%.
- PFG guarantees that $G(\mathbf{s})$ is computationally indistinguishable to a so-called, *flawed-smudging distribution* $\mathbf{Y} \leftarrow \mathcal{Y}$, satisfying that given $\mathbf{Y} + \mathbf{e}$, the values of $\mathbf{e}$ at a few $o(\lambda)$ coordinates are revealed, while the values at the rest coordinates are hidden.

We elaborate on the security definitions of these generators, and possible instantiations, in Section 2.

Hardness of polynomials over the reals. The security of our candidate ΔRGs/PFGs crucially relies on the hardness of solving certain over-determinined systems of degree-3 polynomial equations over the reals, and a LWE leakage assumption. Solving systems of polynomials over the reals has been studied by mathematicians, scientists, and engineers for hundreds of years. This is precisely why we are taking this approach: we want to relate $i\mathcal{O}$ to simple-to-state problems related to areas of mathematics with long histories of study. Aside from that, our work also fundamentally diversifies the kinds of assumptions from which $i\mathcal{O}$ can be constructed.

In Section 2.4, we describe specific candidates suggested in follow-up work by [BHJ$^+$19] that were inspired by the hardness of RANDOM 3-SAT. We hope that our work will motivate further cryptanalytic study of simple pseudorandom objects.

Using respecitively ΔRG and PFG, we show how to construct $i\mathcal{O}$ without multilinear maps of degree ≥ 3 in two concurrent works [AJS18, LM18]. Next, we describe the results in each work slightly more formally.

Results in AJS in more detail. AJS constructs $i\mathcal{O}$ based on bilinear maps, LWE, ΔRG, and block-locality 3 PRG. For the latter, in fact AJS require only a weakened forms of 3-blockwise-local PRGs [LT17] where efficient adversaries fail to distinguish the the PRG output distribution from the uniformly random distribution with some polynomial probability[9].

Theorem 1 (AJS Main Theorem, Informal). *For every constants c, there is a construction of indistinguishability obfuscation for all polynomial-sized circuits from,*

- $\left(1 - \frac{1}{\lambda^c}\right)$*-indistinguishable perturbation-resilient generators with aforementioned structure and security against sub-exponential size adversaries,*
- $\frac{1}{2\lambda^c}$*-indistinguishable three-block-local pseudorandom generators [LT17] with polynomial stretch and security against sub-exponential size adversaries,*
- *learning with errors secure against sub-exponential size adversaries, and*
- *assumptions on bilinear maps secure against sub-exponential size adversaries (that hold unconditionally in the generic bilinear map model).*

Here κ-indistinguishability refers to security where the distinguishing advantage of such adversaries is bounded by κ. Thus, standard security would be $\mathsf{negl}(\lambda)$-security, where negl is a negligible function. In contrast $(1 - p)$-security allows for an adversary that fails to distinguish only with probability p.

Along the way to proving the result above, AJS also obtains a securty amplification theorem for functional encryption:

Theorem 2 (AJS security amplification theorem, informal). *Assume there exists a constant c > 0, and*

- $(1 - 1/\lambda^c)$*-indistinguishable sublinearly compact secret key FE schemes for polynomial size circuits of depth λ, and*
- *learning with errors secure against sub-exponential size adversaries.*

There exists sublinearly compact secret key FE schemes for polynomial size circuits of depth λ with $\mathsf{negl}(\lambda)$-indistinguishability.

Note that the amplification theorem above relies only on subexponential LWE, and no new assumptions. Moreover, if we assume the underlying FE schemes to be secure against subexponential size, then the resulting schemes satisfy subexponential security. Please refer [AJS18] for a complete formulation.

Results in LM in more detail. LM constructs $i\mathcal{O}$ based on bilinear maps, LWE, PFGs, and constant block-local PRGs.

Theorem 3 (LM Main Theorem, informal). *There is a construction of indistinguishability obfuscation for all polynomial-sized circuits from,*

---

[9] There is be a tradeoff between how much AJS can weaken the indistinguishability requirements of the ΔRG and the 3-block-local PRG.

- *pseudo flawed-smudging generators with aforementioned structure and security against sub-exponential size adversaries,*
- *constant-block-local pseudorandom generators [LT17] with mild structural properties described in Remark 1, and security against sub-exponential size adversaries,*
- *learning with errors secure against sub-exponential size adversaries, and*
- *the SXDH assumption on bilinear maps secure against sub-exponential size adversaries.*

Remark 1. The block-local PRGs used in LM map n bits to $n^{1+\alpha}$ bits for an arbitrarily small constant α, where every PRG is defined by a predicate P and an input-output dependency graph G, such that the i'th output bit $y_i = P(\mathsf{Seed}_{G(i)})$ is computed by evaluating the predicate P on a subset of seed bits $\mathsf{Seed}_{G(i)}$ specified by $G(i)$. LM requires the output locality (i.e., $\max_i |G(i)|$) to be a constant, and the input locality (i.e., the maximal number of output bits that an input bit influences) to be bounded by $o(n^{1-\alpha})$. Most candidate constant-locality PRGs [Gol00, MST03, OW14, AL16] satisfy these structural properties. In particular, the input-output dependency graph is often chosen at random in which case the input locality is indeed bounded by $o(n^{1-\alpha})$. The security of local PRGs, especially ones with large constant locality, has been studied extensively, for instance in [CM01, MST03, CEMT09, BQ12, OW14, AL16].

Partially Hiding Functional Encryption. In order to evaluate ΔRGs/PFGs using bilinear map, we develop the primitive of Partially Hiding Functional Encryption schemes (PHFE), introduced under the name 3-restricted FE by [AJS18]. The notion of PHFE is a natural modification of partially-hiding Predicate Encryption (PE) of [GVW15] by strengthening the security requirement from that of PE to FE. PHFE schemes can evaluate functions of the form $g(\mathbf{x}, \mathbf{y})$ and guarantee that ciphertexts and secret keys reveal only the outputs and part of its input $\mathbf{x}$, referred to as the *public input*, while hiding the remaining part $\mathbf{y}$, referred to as the *private input*. Partially-hiding FE naturally interpolates attribute-based encryption and functional encryption: if the public input $\mathbf{x}$ is empty, it is equivalent to functional encryption, and if g is such that it outputs $\mathbf{y}$ when some predicate on $\mathbf{x}$ evaluates to 1, then it corresponds to attribute-based encryption.

In the literature, there are constructions of secret-key FE schemes for quadratic polynomials from bilinear map groups [Lin17, BCFG17]. In AJS and LM, we extend these constructions to allow for additional linear computation on a public input.

Theorem 4 (PHFE in AJS and LM, Informal). *There are constructions of secret-key partially-hiding FE schemes for computing multilinear cubic polynomials $g(\mathbf{x}, (\mathbf{y}, \mathbf{z}))$ over $\mathbb{Z}_p$ with polynomially bounded outputs and $\mathbf{x}$ as the public input, from bilinear pairing groups of order p. The schemes have linear encryption time $\mathrm{poly}(\lambda)N$ in the input length $N = \max(|\mathbf{x}|, |\mathbf{y}|, |\mathbf{z}|)$.*

The constructions of PHFE in AJS and LM differ in details. The scheme originally developed in AJS, referred to as 3-restricted FE there, follows the semi-functional

FE framework and is based on assumptions over bilinear maps that hold unconditionally in the generic bilinear map model. The scheme subsequently developed in LM, referred to as degree-(1,2) PHFE there, satisfies simulation security for one ciphertext (meaning that the outputs evaluated on one encryption input can be programmed) and is based on SXDH.

Finally, we mention that in followup works, our approach has been extended to *i)* use ΔRGs/PFGs implementable by polynomials of any *constant*-degree [JLMS19], *ii)* remove the need for block-local PRGs completely [JLS19], and *iii)* construct PHFE supporting NC_1 public computation and degree 2 private comptuation [JLS19].

1.1 History

We provide a timeline describing how the results were conceived, to clarify how this line of work has developed.

06/17/2018: [AJS18] received by Eprint (2018/615).

[AJS18] introduced ΔRGs, 3-restricted FE, and a new general FE amplification theorem.

Historical notes: Earlier weaker versions of [AJS18] were submitted to EC 2018 (on 9/19/2017) and Crypto 2018 (on 2/13/2018). These earlier versions contained the notions of 3-restricted FE, and Tempered Cubic Encoding. However, they did not contain either the notion of ΔRG nor the FE amplification theorem. The authors of [AJS18] were not aware of the relevant concurrent work by [Agr18a] or [LM18] until seeing Eprint papers appear.

06/17/2018: [Agr18a] received by Eprint (2018/633).

To hide decryption noises, [Agr18a] introduced different notions of (smudging) noise generators, which all *perfectly* hides the noises. Hence [Agr18a] did not develop any FE security amplification technique. In terms of instantiation, [Agr18a] proposed using MQ or 2 block-local PRG as degree 2 candidates and used off-the-shelf deg 2 FE to evaluate them. [Agr18a] contains a gap in the construction: It proposes to use known deg 2 FE to compute the noise generator. Known deg 2 FE restricts the outputs of the noise generator to be poly-large. On the other hand, [Agr18a] needs the noise generator to perfectly hide the HE decryption noise e, which requires the outputs to be super-poly large. (Note: this is why [AJS18]'s ΔRG and [LM18]'s PFG only provide weak guarantees. This allows for having poly-large outputs, but opens many challenges in order to deal with the weak guarantees.)

07/02/2018: [LM18] received by Eprint (2018/646).

In Aug 2017, Lin discussed with Agrawal about her ideas and Agrawal shared a manuscript. After the discussion, Agrawal and Lin proceeded independently. Since the shared manuscript has large overlap with the later posted [Agr18a], [LM18] simply treats entire [Agr18a] as prior work for clarity.

Prior to posting, [LM18] has developed for over a year. [LM18] introduced the notion of Pseudo Flawed-smudging Generator (PFG) and the leakage-based security amplification technique. They analyzed PFG properties and proposed using deg 2 polynomials sampled from a special distribution as the candidates.

07/08/2018: [AJS18] updated on Eprint (2018/615).
Added explicit degree 3 ΔRG candidate and associated explicit ΔRG assumption.

08/17/2018: [Agr18a] updated on Eprint (2018/633).
[Agr18a] cites [AJS18] for fixing the aforementioned gap. This means using the notion of ΔRG and the FE security amplification theorem of [AJS18].

08/19/2018: [BHJ+19] announced at "Beyond Crypto" workshop at CRYPTO 2018.
This work gave empirical and theoretical evidence of polynomial-time attacks on all known explicit degree-2 candidates considered in [AJS18, Agr18a, LM18]. It is explicitly noted that the attacks do not extend to the degree 3 ΔRG candidate of [AJS18].

10/4/2018: [JS18], [BHJ+19] submitted to Eurocrypt 2019.
[JS18] showed how to construct constant-restricted FE for any constant (i.e., (deg-O(1), deg 2)-PHFE) assuming SXDH. This enables using constant degree candidates, for any constant. [JS18] is clearly marked as a follow-up work to [AJS18, Agr18a, LM18].

10/9/2018: The second version of [LM18] was updated on Eprint (2018/646). Added construction of (deg 1, deg 2)-PHFE, which is a variant of 3-restricted FE, and proposed to use the degree 3 candidate of [AJS18] as candidate PFGs, which are not subject to [BHJ+19] attacks. This updated [LM18] clearly cites [AJS18] for this candidate and the idea of using weak deg-3 FE to evaluate it. However, note that this just replaces the previous deg 2 candidate and deg 2 FE in [LM18], which are very simple and not the main technical contributions of [LM18].

In this update, there is also a construction of PHFE able to handle public computation of poly degree, but subject to certain size constraints. This construction does not appear in this current paper for two reasons: 1) Lin and Matt were added as authors to [JS18] in credit for this concurrent PHFE construction, and 2) it is later superseded by a full (NC1, deg 2) PHFE construction in a follow-up [JLS19].

10/11/2018: [JS18] received by Eprint (2018/973).

02/01/2019: [JS18], [BHJ+19] accepted at Eurocrypt 2019.
The authors of [JS18] emailed Chairs to add Lin and Matt as authors, resulting in publication [JLMS19]. The paper [JLMS19] is clearly marked as a follow-up

work to [AJS18, Agr18a, LM18].

1.2 Comparison of Techniques

We provide a detailed comparison of the works of [AJS18, LM18, Agr18b, BHJ$^+$19, JLMS19].

Comparison of the works of [AJS18] and [LM18]. We first start by comparing the notions of PFGs and ΔRGs. Both notions are geared for the purpose of generating a smudging noise $\mathbf{Y}$ to hide a small polynomially bounded noise $\mathbf{e}$, however, with different guarantees. The output $\mathbf{O}$ of PFGs is *computationally indistinguishable* to flawed smudging noises $\mathbf{Y}$ such that $(\mathbf{e}, \mathbf{Y} + \mathbf{e})$ and $(\mathbf{e}', \mathbf{Y} + \mathbf{e})$ are statistically close with probability δ. On the other hand, the output $\mathbf{O}$ of ΔRGs directly guarantees that $(\mathbf{e}, \mathbf{O} + \mathbf{e})$ and $(\mathbf{e}', \mathbf{O} + \mathbf{e})$ are computationally indistinguishable up to advantage $1 - \delta$. Furthermore, in the good case with probability δ, the output of PFGs may still reveal $\mathbf{e}$ at a few coordinates (i.e., $\mathbf{e}$ and $\mathbf{e}'$ agree at a few coordinates), whereas ΔRG ask for weak indistinguishability between the two cases (i.e. $\mathbf{e}$ and $\mathbf{e}'$).

Besides the use of different weak notions of randomness generators, other differences between [AJS18] and [LM18] include: *i)* [LM18] rely on constant-locality PRGs with mild structural properties, while [AJS18] use block-locality 3 PRGs. *ii)* [AJS18] first showed security in the generic bilinear map model, subsequently [LM18] relied on the SXDH assumption over bilinear pairing groups.

In terms of techniques, both works start with constructing some weak notions of FE: [LM18] construct FE for constant-degree polynomials that may leak a small portion of the input, whereas [AJS18] construct FE for degree 3 polynomials that bounds the adversarial advantage only by $1 - 1/\text{poly}(\lambda)$. Both works then design different methods to amplify their respective weak FE to full-fledged FE. The amplification techniques are similar in parts, for instance, both works use threshold FHE, but also have differences, for instance, while [LM18] relies on the use of random permutations and a careful analysis to ensure that the effect of compromising a few bits of the seed of a constant-locality PRG can be "controlled". On the other hand, [AJS18] use techniques from the dense model theorem to give a general security amplification for Functional Encryption with weak distinguishing advantage, into Functional Encryption satisfying the standard notion of security.

Comparison of [AJS18, LM18] with the work of Agrawal [Agr18a]. Following [AR17], to obtain compact ciphertexts, Agrawal [Agr18b] (*as mentioned in the timeline, an early version of [Agr18a] was shared by the author of [Agr18b] with the authors of [LM18]*) proposed the approach of using a noise generator to generate $\mathbf{Y}$. As an abstraction of that, they introduced the notion of *noisy linear functional encryption* that adds the smudging noises $\mathbf{Y}$ to the outputs. The noise generator in [Agr18b] is able to produce super-polynomially large smudging noises, and they propose a constant degree FE scheme supporting

super-polynomially large outputs from a new assumption on NTRU Rings. The works of [AJS18] and [LM18] explore what happens when $\mathbf{Y}$ is polynomially bounded and $\mathbf{e}$ may be leaked, which allows us to use FE schemes supporting only polynomially large outputs from multilinear maps.

Subsequent to [AJS18], Agrawal notes that their construction is compatible with the approach of [AJS18] using ΔRG with polynomially large outputs and weak security, and later amplifying the security of FE in a black-box way. Thus, the construction in the updated version can use known constructions of FE schemes with restricted output size.

Comparison with the work by Jain, Lin, Matt and Sahai [JLMS19]. As a follow-up to [AJS18, LM18, Agr18a], Jain, Lin, Matt and Sahai [JLMS19] construct FE schemes for degree $d + 2$ functions multilinear in their inputs $\mathbf{x}_1, \ldots, \mathbf{x}_d, \mathbf{y}, \mathbf{z}$, where $\mathbf{x}_1, \ldots, \mathbf{x}_d$ are public, $\mathbf{y}$ and $\mathbf{z}$ are private, and d can be any constant. They further improve upon [AJS18] by only relying on the SXDH assumption instead of the generic bilinear map model. Moreover, their work provides new candidates of ΔRGs that can be computed by their FE schemes. Similar to [AJS18, LM18], their candidates hide the public inputs as noises in LWE samples.

Comparison of [AJS18, LM18] and [GKP$^+$13, GVW15, BTVW17, AR17]. Both the works of [AJS18, LM18] use a homomorphic encryption scheme (HE) in conjunction with the newly introduced pseudorandom generators to construct FE. This approach of using a homomorphic encryption scheme to construct FE is not new has already been explored in several works [GVW12, GKP$^+$13, GVW15, BTVW17, AR17, Agr18a]. The challenges to build FE from HE are twofold: *1) privacy*—decrypt a ciphertext $\mathsf{CT}_{f,x}$ encrypting an output $f(x) = \mathbf{y}$ securely revealing only $\mathbf{y}$, and *2) integrity*—enforce that only ciphertexts for "legitimate" functions f (ones for which secret keys are generated) can be decrypted. Below, we briefly discuss how this approach was adopted in previous works.

The work of Goldwasser et al. [GKP$^+$13] use the above template to build a single-functional encryption scheme. They use an attribute-based encryption scheme to ensure integrity and garbled circuits to ensure privacy. Then they combine both these tools along with HE to achieve their result.

Gorbunov, Vaikuntanathan, and Wee [GVW15], also using the above approach, construct a predicate encryption scheme based on learning with errors; recall that predicate encryption is a weaker form of functional encryption. They propose a novel primitive, called partial-hiding predicate encryption scheme and then combine it with HE to obtain a predicate encryption scheme. Their notion of partial-hiding predicate encryption scheme incorporates both the privacy and the integrity properties. In terms of techniques, the starting point to their construction of partial-hiding predicate encryption scheme is the observation that the HE decryption corresponds to computing an inner product followed by a threshold function. Moreover, there are lattice-based constructions of predicate encryption schemes for threshold of inner product [AFV11, GMW15]. They then propose a novel method to combine a lattice-based predicate encryption for threshold of inner product with a lattice-based attribute-based encryption scheme

10

to achieve a partial-hiding predicate encryption scheme. Natural attempts to extend their construction to achieve functional encryption have been shown to be broken [Agr17a].

In [AR17], to ensure privacy of HE decryption, they use an FE scheme to perform linear HE half-decryption and add super-polynomially large smudging noises $\mathbf{Y}$ to hide the decryption noise $\mathbf{e}$. In their scheme, the smudging noises $\mathbf{Y}$ are sampled and encoded into the ciphertext. As a result, the ciphertext size grows with the output length of the computation, which is non-compact. In addition, they also developed a new approach to ensure integrity. Instead of relying on primitives like attribute based encryption or PHFE to ensure integrity as in [GVW12, GKP$^+$13, GVW15], they employ a special HE scheme whose decryption equation has the form $\mathbf{y}+\mathbf{e} = \mathbf{c}_f - \mathbf{A}_f\mathbf{s}$, where $\mathbf{A}_f$ depends only on the public and reusable random matrix $\mathbf{A}$ in LWE samples and the evaluated function f. Thus, to ensure integrity, it suffices to enforce that only linear functions $\mathbf{A}_f\mathbf{s}$ for legitimate f can be evaluated on $\mathbf{s}$. The work of [LM18] follows their approach for integrity. The work of [AJS18], however, takes a different path, by introducing the notion of 3-restricted FE (that we call partially hiding functional encryption here).

1.3 Open Questions

Our work opens many interesting questions. First, we call for more study of the candidate ΔRGs/PFGs. Studying their security as well as finding new candidates may build interesting connection with algorithm and complexity theory as already demonstrated in the attack by [BHJ$^+$19] using SOS algorithms.

Secondly, can we further strengthen the construction of FE or $i\mathcal{O}$ in order to further weaken the requirements on the structure and security of ΔRGs/PFGs? Follow-up works show how to construct PHFE schemes that can perform constant-degree [JLMS19] computation or even up to NC_1 computation [JLS19] in the public input, instead of just linear (still quadratic in the private input). Such scheme allows for having more candidate ΔRGs/PFGs.

Thirdly, the reason that we can only work with polynomially bounded smudging noises is because we do not have constant-degree FE schemes that support super-polynomially large outputs from multilinear maps and/or standard assumptions. For instance, can we build a quadratic FE scheme for super-polynomially large outputs from standard assumptions? That would lead to a significant simplification of our construction of NC^1-FE as there would be no more leakage.

2 New PRG Assumptions

This section is organized as follows. In Section 2.1 we define the notion of perturbation resilient generator ΔRG proposed by [AJS18]. In Section 2.2 we define pseudo-flawed smudging generators (PFGs) proposed by [LM18]. Then, in Section 2.3 we give an algorithmic framework to realise ΔRG and PFG. Both of them are PRGs which has seed consisting of one public input and two secret

input. These PRGs evaluate degree-3 multilinear polynomials over $\mathbb{Z}_\mathbf{p}$ over these inputs. In the same section, we give an intuition as to why this structure can be realised using bilinear maps. In Section 2.4 we give candidate polynomials which can be used to instantiate these primitives. In Section 2.5, we illustrate a *single* assumption which will imply the notion of a perturbation resilient generator sufficient to build $i\mathcal{O}$ [AJS18]. In Section 2.6 we present the state of art in cryptanalysis of the candidate polynomials.

2.1   Perturbation Resilient Generator

A perturbation-resilient generator, denoted by $\varDelta$RG, consists of the following algorithms:

- **Setup,** Setup$(1^\lambda, 1^n, B)$: On input security parameter λ, the length parameter n and a polynomial $B = B(\lambda)$, it outputs a seed Seed and public parameters pp.
- **Evaluation,** Eval(pp, Seed): It takes as input public parameters pp, seed Seed and outputs a vector $(h_1, ..., h_\ell) \in \mathbb{Z}^\ell$. The parameter ℓ is defined to be the stretch of $\varDelta$RG.

The following properties are associated with a $\varDelta$RG scheme.

Efficiency: The following conditions need to be satisfied.

- The time taken to compute Setup$(1^\lambda, 1^n, B)$ is $n \cdot \text{poly}(\lambda)$ for some fixed polynomial poly.
- For all $i \in [\ell]$, $|h_i| = \text{poly}(\lambda, n)$. That is, the norm of each component h_i in $\mathbb{Z}$ is bounded by some polynomial in λ and n.

Perturbation Resilience: For every polynomial $B(\lambda)$, for every large enough polynomial $n = n(\lambda)$ and for all large enough λ, the following holds: for every $a_1, ..., a_\ell \in \mathbb{Z}$, with $|a_i| \leq B(\lambda)$, we have that for any distinguisher D of size 2^λ,

$$\left| \Pr_{x \xleftarrow{\$} \mathcal{D}_1} [1 \leftarrow D(x)] - \Pr_{x \xleftarrow{\$} \mathcal{D}_2} [1 \leftarrow D(x)] \right| < 1 - 1/\lambda,$$

where the sampling algorithms of $\mathcal{D}_1$ and $\mathcal{D}_2$ are defined as follows:

- Distribution $\mathcal{D}_1$: Compute $(\text{pp}, \text{Seed}) \leftarrow$ Setup$(1^\lambda, 1^n, B)$ and $(h_1, ..., h_\ell) \leftarrow$ Eval(pp, Seed). Output $(\text{pp}, h_1, ..., h_\ell)$.
- Distribution $\mathcal{D}_2$: Compute $(\text{pp}, \text{Seed}) \leftarrow$ Setup$(1^\lambda, 1^n, B)$ and $(h_1, ..., h_\ell) \leftarrow$ Eval(pp, Seed). Output $(\text{pp}, h_1 + a_1, ..., h_\ell + a_\ell)$.

Note that as is, we are not able to use the notion of a $\varDelta$RG to construct iO. We now define the notion of a perturbation-resilient generator implementable by a three-restricted FE scheme ($3\varDelta$RG for short). This notion turns out to be useful for our construction of iO.

ΔRG implementable by Three-Restricted FE. A ΔRG scheme implementable by Three-Restricted FE (3ΔRG for short) is a perturbation resilient generator with some additional structural properties. We describe syntax again for a complete specification.

- Setup$(1^\lambda, 1^n, B) \to (\mathsf{pp}, \mathsf{Seed})$. The setup algorithm takes as input a security parameter λ, the length parameter 1^n and a polynomial $B = B(\lambda)$ and outputs a seed Seed and public parameters pp. Here, Seed $=$ (Seed.pub, Seed.priv(1), Seed.priv(2)) is a vector on $\mathbb{Z}_{\mathbf{p}}$ for a modulus $\mathbf{p}$, which is also the modulus used in three-restricted FE scheme. There are three components of this vector, where one of the component is public and two components are private, each in $\mathbb{Z}_{\mathbf{p}}^{n\,\mathrm{poly}(\lambda)}$. Also each part can be partitioned into sub-components as follows. Seed.pub $=$ (Seed$_{\mathsf{pub},1}$, ..., Seed$_{\mathsf{pub},n}$), Seed.priv(1) $=$ (Seed$_{\mathsf{priv}(1),1}$, ..., Seed$_{\mathsf{priv}(1),n}$) and Seed.priv(2) $=$ (Seed$_{\mathsf{priv}(2),1}$, ..., Seed$_{\mathsf{priv}(2),n}$). Here, each sub component is in $\mathbb{Z}_{\mathbf{p}}^{\mathrm{poly}(\lambda)}$ for some fixed polynomial poly independent of n. Also, pp $=$ (Seed.pub, $q_1, .., q_\ell$) where each q_i is a cubic multilinear polynomial described in the next algorithm. We require syntactically there exists two algorithms SetupSeed and SetupPoly such that Setup can be decomposed follows:
    1. SetupSeed$(1^\lambda, 1^n, B) \to \mathsf{Seed}$. The SetupSeed algorithm outputs the seed.
    2. SetupPoly$(1^\lambda, 1^n, B) \to q_1, ..., q_\ell$. The SetupPoly algorithm outputs $q_1, .., q_\ell$.
- Eval$(\mathsf{pp}, \mathsf{Seed}) \to (h_1, ..., h_\ell)$, evaluation algorithm output a vector $(h_1, ..., h_\ell) \in \mathbb{Z}^\ell$. Here for $i \in [\ell]$, $h_i = q_i(\mathsf{Seed})$ and ℓ is the stretch of 3ΔRG. Here q_i is a cubic polynomial which is multilinear in public and private components of the seed.

The security and efficiency requirements are same as before.

Remark 2. To construct iO we need the stretch of 3ΔRG to be equal to $\ell = n^{1+\epsilon}$ for some constant $\epsilon > 0$.

We can construct 3ΔRG from a succintly stated, instance independent and a falsifiable assumption stated in Section 2.5.

2.2 Pseudo-Flawed Smudging Generators

In this section, we first define what it means for a distribution over $\mathbb{Z}^\ell$ to be smudging and flawed-smudging, and then introduce pseudo flawed-smudging generators.

First, the distribution of a random variable X is smudging if the statistical distance between X and $X + e$ is small for all e with bounded magnitude.

Definition 1 (Smudging distributions). *Let ℓ be a positive integer, let $\varepsilon \in [0, 1]$, and let B either be a positive integer or an ℓ-dimensional vector of positive integers. We say a distribution $\mathcal{X}$ over $\mathbb{Z}^\ell$ is (B, ε)-smudging if for $X \leftarrow \mathcal{X}$ and for all B-bounded $e \in \mathbb{Z}^\ell$, we have $\delta(X, X + e) \leq \varepsilon$.*

We next define distributions obtained by fixing some positions in the output of a distribution. This will be used for defining flawed-smudging distributions.

Definition 2 (Bit-fixing distributions). *Let $\mathcal{D}$ be a distribution over strings in Δ^ℓ for some set Δ and some integer ℓ. Let $I \subseteq [\ell]$ be a set of indices, and x an arbitrary string in $\Delta^{|I|}$. Define $\mathcal{D}|_{x,I}$ to be the distribution of sampling $\overline{x}$ from $\mathcal{D}$ conditioned on $\overline{x}_I = x$. For convenience, we sometimes also write I as its characteristic vector v, where $v_i = 1$ iff $i \in I$.*

We say that $\mathcal{D}$ is bit-fixing efficiently samplable if $\mathcal{D}|_{x,I}$ is efficiently samplable for any x, I.

We now define flawed-smudging distributions. On a high level, the distribution of X is flawed-smudging for a random variable E if there are a few "bad" coordinates such that $X + E$ "hides" E at all coordinates that are not bad. This means, given $X + E$ and which coordinates are bad, one cannot distinguish E from $\overline{E}$, where $\overline{E}$ is a fresh sample conditioned on agreeing with E on the bad coordinates.

Definition 3 (Flawed-smudging distributions). *Let ℓ be a positive integer and let $\mathcal{X}$ and $\mathcal{E}$ be distributions over $\mathbb{Z}^\ell$. Further let $K \in \mathbb{N}$ and $\mu \in [0,1]$. We say that $\mathcal{X}$ is (K, μ)-flawed-smudging for $\mathcal{E}$ if there exist randomized predicates $\left\{ \mathrm{BAD}_i \colon \mathbb{Z}^{\ell+1} \to \{0,1\} \right\}_{i \in [\ell]}$ such that the following two distributions are identical:*

$$\mathcal{D}_1 = \left\{ \begin{array}{c} E \leftarrow \mathcal{E} \\ X \leftarrow \mathcal{X} \\ \mathrm{bad} = \left(\mathrm{bad}_i \leftarrow \mathrm{BAD}_i(E_i, X)\right)_{i \in [\ell]} \end{array} \colon (E,\ X + E,\ \mathrm{bad}) \right\},$$

$$\mathcal{D}_2 = \left\{ \begin{array}{c} E \leftarrow \mathcal{E} \\ X \leftarrow \mathcal{X} \\ \mathrm{bad} = \left(\mathrm{bad}_i \leftarrow \mathrm{BAD}_i(E_i, X)\right)_{i \in [\ell]} \\ \overline{E} \leftarrow \mathcal{E}|_{E_{\mathrm{bad}}, \mathrm{bad}} \end{array} \colon \left(\overline{E},\ X + E,\ \mathrm{bad}\right) \right\},$$

and in addition, with probability at least $1 - \mu$, the 1-norm of bad *is bounded by $|\mathrm{bad}|_1 \leq K$.*

We say the distribution $\mathcal{X}$ is (K, μ)-flawed-smudging for B-bounded distributions if it is (K, μ)-flawed-smudging for every B-bounded distribution $\mathcal{E}$, where B can either be a positive integer or a vector in $\mathbb{Z}^\ell$.

Remark 3. A more direct generalization of the definition of smudging distributions (see Definition 1) would be that for all e, the distribution of $X + e$ is equal (or statistically close) to the distribution of Y, where $Y_i = X_i + e_i$ for all bad i, and $Y_i = X_i$ for non-bad i. This is, however, not sufficient for our purposes: We need that no information about the non-bad coordinates is leaked. While X_i itself does not leak anything about e_i, the fact that i is not a bad coordinate can leak something about e_i, since the predicate BAD depends on e_i. Definition 3 resolves this issue by sampling the non-bad coordinates freshly after sampling bad.

Pseudo flawed-smudging generators. We now define pseudo flawed-smudging generators (PFGs). A PFG is a distribution of efficiently computable functions and seeds for which the output of the functions is indistinguishable from a flawed-smudging distribution.

Definition 4. *(Pseudo Flawed-Smudging Generator) Let n, m, K, B be polynomials. A family of (K, μ)-pseudo flawed-smudging generators $((K, \mu)$-PFG) for B-bounded distributions is an ensemble of distributions $\mathcal{PFG} = \{\mathcal{PFG}_\lambda\}_{\lambda \in \mathbb{N}}$ satisfying the following properties:*

Syntax: *For every $\lambda \in \mathbb{N}$, every $(\mathrm{PFG}, \mathcal{D}^{sd})$ in the support of $\mathcal{PFG}_\lambda$ defines a function $\mathrm{PFG} \colon \mathbb{Z}^{n(\lambda)} \to \mathbb{Z}^{m(\lambda)}$ and a distribution $\mathcal{D}^{sd}$ over seeds.*

Efficiency: *There is a uniform Turing machine M satisfying that for every $\lambda \in \mathbb{N}$, every $(\mathrm{PFG}, \mathcal{D}^{sd}) \in \mathrm{Support}(\mathcal{PFG}_\lambda)$ and $\mathsf{Seed} \in \mathrm{Support}(\mathcal{D}^{sd})$, $M(\mathrm{PFG}, \mathsf{Seed})$ runs in time $\mathrm{poly}(\lambda)$ and we have $M(\mathrm{PFG}, \mathsf{Seed}) = \mathrm{PFG}(\mathsf{Seed})$. Furthermore, $\mathcal{PFG}$ and all $\mathcal{D}^{sd}$ in the support of $\mathcal{PFG}_\lambda$ are efficiently samplable.*

(K, μ)-pseudo-flawed-smudging for B-bounded distributions: *There exists an ensemble $\{\mathcal{X}_\lambda\}$ of distributions, such that the distribution $\mathcal{X}_\lambda$ is $(K(\lambda), \mu(\lambda))$-flawed-smudging for all $B(\lambda)$-bounded distributions, and the following ensembles are μ-indistinguishable:*

$$\left\{ (\mathrm{PFG}, \mathcal{D}^{sd}) \leftarrow \mathcal{PFG}_\lambda; \mathsf{Seed} \leftarrow \mathcal{D}^{sd} : (\mathrm{PFG}, \mathrm{PFG}(\mathsf{Seed})) \right\}_{\lambda \in \mathbb{N}},$$

$$\left\{ (\mathrm{PFG}, \mathcal{D}^{sd}) \leftarrow \mathcal{PFG}_\lambda; X \leftarrow \mathcal{X}_\lambda : (\mathrm{PFG}, X) \right\}_{\lambda \in \mathbb{N}}.$$

Degree 3 PFG with Partial Public Input: As mentioned in the introduction and as described w.r.t. $\Delta\mathsf{RG}$, it suffices if our PFG has the simple structure that every funtion PFG sampled from $\mathcal{PFG}_\lambda$ is a degree 2 polynomial over $\mathbb{Z}_p$, where p is a modulus that eventually matches the modulus that our PHFE supports, which in turn is the modulus associated with the bilinear maps. However, so far, we do not know how to instantiate a truly degree 2 PFG. Instead, we can work with the following slightly weaker structure, where the PFG is a degree 3 multilinear polynomial, and the first input vector can be made public, more specifically:

Structure: For every λ, every $(\mathrm{PFG}, \mathcal{D}^{sd}) \in \mathcal{PFG}_\lambda$ satisfies that $\mathcal{D}^{sd}$ is a distribution over $(\mathbf{x}, \mathbf{y}, \mathbf{z}) \in \mathbb{Z}_p^3$ for some modulus p, and $\mathrm{PFG}(\mathbf{x}, \mathbf{y}, \mathbf{z})$ is a multilinear degree 3 polynomial over $\mathbb{Z}_p^3$.

Security with partial public input: The security in Definition 4 is strengthened so that the following distributions are indistinguishable:

$$\left\{ (\mathrm{PFG}, \mathcal{D}^{sd}) \leftarrow \mathcal{PFG}_\lambda; \mathsf{Seed} = (\mathbf{x}, \mathbf{y}, \mathbf{z}) \leftarrow \mathcal{D}^{sd} : (\mathrm{PFG}, \mathbf{x}, \mathrm{PFG}(\mathsf{Seed})) \right\}_{\lambda \in \mathbb{N}},$$

$$\left\{ (\mathrm{PFG}, \mathcal{D}^{sd}) \leftarrow \mathcal{PFG}_\lambda; \mathsf{Seed} = (\mathbf{x}, \mathbf{y}, \mathbf{z}) \leftarrow \mathcal{D}^{sd}; X \leftarrow \mathcal{X}_\lambda : (\mathrm{PFG}, \mathbf{x}, X) \right\}_{\lambda \in \mathbb{N}}.$$

Weaker variant: Flawed-smudging with $1/\mathrm{poly}(\lambda)$ probability. In the full version [LM18], we show how to further weaken the requirements on PFGs. Roughly speaking, the PFG outputs are indistinguishable to a flawed-smudging distribution only with some $1/\mathrm{poly}(\lambda)$ probability. We show that using essentially the same technique for handling the partial hiding guarantee of PFG can also be used to handle this weakening. We omit details here; see [LM18] for more details.

Properties of (Flawed-)Smudging Distributions In the full version [LM18], we prove some properties of smudging and of flawed-smudging distributions. More specifically, we show the following:

- Polynomially bounded distributions cannot be smudging with negligible ε. More precisely, if $\mathcal{X}$ is B-bounded and (B', ε)-smudging, then $\varepsilon \geq \frac{1}{2B+1}$.
- Adding independent values preserves the (flawed-)smudging property, i.e., if X and Y are independent and the distribution of X is (flawed-)smudging, then the distribution of $X+Y$ is (flawed-)smudging with the same parameters.
- Probabilistically mixing (flawed-)smudging distributions yields a (flawed-)smudging distribution. That is, if the distributions of X_i are (B, ε_i)-smudging (or (K, μ_i)-flawed-smudging) and $\alpha_i \in [0,1]$ such that $\sum_i \alpha_i = 1$, then the distribution of X with $\Pr[X = x] = \sum_i \alpha_i \Pr[X_i = x]$ is $(B, \sum_i \alpha_i \varepsilon_i)$-smudging (or $(K, \sum_i \alpha_i \mu_i)$-flawed-smudging).
- The joint distribution of mutually independent smudging distributions is flawed-smudging. More precisely, we show that if $\mathcal{X}$ is a distribution over $\mathbb{Z}^\ell$ such that for $(X_1, \ldots, X_\ell) \leftarrow \mathcal{X}$, $X_1, \ldots, X_\ell$ are mutually independent and the distribution of each X_i is (B, ε)-smudging for $\varepsilon \leq \frac{K+1}{22\ell \cdot (2B+1)}$, then $\mathcal{X}$ is $(K, 2^{-K})$-flawed-smudging for B-bounded distributions.
- The product of flawed-smudging distributions is flawed-smudging. That is, for distributions $\mathcal{X}^{(1)}$ and $\mathcal{X}^{(2)}$ such that $\mathcal{X}^{(i)}$ is $(K^{(i)}, \mu^{(i)})$-flawed-smudging, we have that $\mathcal{X}^{(1)} \times \mathcal{X}^{(2)}$ is $(K^{(1)} + K^{(2)}, \mu^{(1)} + \mu^{(2)})$-flawed-smudging.
- If the distribution of X is flawed-smudging for the distribution of E and $E = E(V)$ is a function of some random vector V such that each coordinate of $E(V)$ depends only on a few coordinates of V, then $E(V) + X$ hides V at all but a few locations.

2.3 Framework for Algorithms of $3\varDelta\mathsf{RG}$ and PFG

We now describe a framework of algorithms that can be used to instantiate $\varDelta\mathsf{RG}$ and PFG. However for the sake of succinctness and clarity we describe it in terms of a perturbation resilient generator $3\varDelta\mathsf{RG}$. For concreteness, we use a large enough prime modulus $\mathbf{p} = O(2^\lambda)$, which is the same as the modulus used by $3-$restricted FE$/(1,2)$-PHFE. Then, let χ be a distribution used to sample input elements over $\mathbb{Z}$. Let Q denote a polynomial sampler. Next we describe the algorithms in terms of χ and Q but give concrete instantiations later in Section 2.4.

- Setup$(1^\lambda, 1^n, B) \to (\mathsf{pp}, \mathsf{Seed})$. Sample a secret $\mathbf{s} \leftarrow \mathbb{Z}_{\mathbf{p}}^{1 \times \mathsf{d}}$ for $\mathsf{d} = \mathrm{poly}(\lambda)$ such that $\mathsf{LWE}_{\mathsf{d}, n \cdot d, \mathbf{p}, \chi}$ holds. Here χ is a bounded distribution with bound $\mathrm{poly}(\lambda)$. Let $\mathcal{Q}$ denote an efficiently samplable distribution of homogeneous degree 3 polynomials (instantiated later). Then proceed with $\mathsf{SetupSeed}$ as follows:
  1. Sample $\mathbf{a}_i \leftarrow \mathbb{Z}_{\mathbf{p}}^{1 \times \mathsf{d}}$ for $i \in [n]$ along with $e_i, y_i, z_i \leftarrow \chi$ for $i \in [n]$.
  2. Compute LWE samples $\mathbf{w}_i = (\mathbf{a}_i, r_i = \langle \mathbf{a}_i, \mathbf{s} \rangle + e_i \mod \mathbf{p})$ for $i \in [n]$.
  3. Output $\mathsf{Seed.pub}(i) = \mathbf{w}_i$ for $i \in [n]$, $\mathsf{Seed.priv}(1, j) = \mathbf{y}_i \otimes (-\mathbf{s}, 1)$ for $j \in [n]$ and $\mathsf{Seed.priv}(2, k) = z_k$ for $k \in [n]$.
- $\mathsf{SetupPoly}$: Now we describe $\mathsf{SetupPoly}$. Fix $\eta = n^{1+\epsilon}$.
  1. Sample polynomials q'_ℓ for $\ell \in [\eta]$ as follows. $q'_\ell(e_1, ..., e_n, y_1, ..., y_n, z_1, ..., z_n) = \Sigma_{\mathbf{I}=(i,j,k)} c_{\mathbf{I}} e_i \cdot y_j \cdot z_k$ where coefficients $c_{\mathbf{I}}$ are bounded by $\mathrm{poly}(\lambda)$. These polynomials $\{q'_\ell\}$ are sampled according to $\mathcal{Q}$
  2. Define q_i be a multilinear homogeneous degree 3 polynomial takes as input $\mathsf{Seed} = (\{\mathbf{w}_i\}_{i \in [n]}, \mathbf{y}'_1, ..., \mathbf{y}'_n, \mathbf{z})$. Then it computes each monomial $c_{\mathbf{I}} e_i y_j \cdot z_k$ as follows and then adds all the results:
     - Compute $c_{\mathbf{I}} \langle \mathbf{w}_i, (-\mathbf{s}, 1) \rangle \cdot y_j \cdot z_k$. This step requires $\mathbf{y}'_i = y_i \otimes (-\mathbf{s}, 1)$ to perform this computation.
  3. Output $q_1, ..., q_\eta$. Observe that $q_i(\mathsf{Seed}) = q'_i(\mathbf{e}, \mathbf{y}, \mathbf{z})$ for all i.
- $\mathsf{Eval}(\mathsf{pp}, \mathsf{Seed}) \to (h_1, ..., h_\eta)$, evaluation algorithm output a vector $(h_1, ..., h_\eta) \in \mathbb{Z}^\eta$. Here for $i \in [\eta]$, $h_i = q_i(\mathsf{Seed})$ and η is the stretch of $3\Delta\mathsf{RG}$. Here q_i is a degree 3 homogenenous multilinear polynomial (defined above) which is degree 1 in public and 2 in private components of the seed.

We prove that the above candidate satisfies the efficiency property of a perturbation-resilient generator.

Efficiency:

1. Note that Seed contains n LWE samples $\mathbf{w}_i$ for $i \in [n]$ of dimension d. Along with the samples, it contains elements $\mathbf{y}'_i = y_i \otimes \mathbf{t}$ for $i \in [n]$ and elements z_i for $i \in [n]$. Note that the size of these elements are bounded by $\mathrm{poly}(\lambda)$ and is independent of n.
2. The values $h_i = q_i(\mathsf{Seed}) = \Sigma_{\mathbf{I}=(i,j,k)} c_{\mathbf{I}} e_i \cdot y_j \cdot z_k$. Since χ is a bounded distribution, bounded by $\mathrm{poly}(\lambda, n)$, and coefficients $c_{\mathbf{I}}$ are also polynomially bounded, each $|h_i| < \mathrm{poly}(\lambda, n)$ for $i \in [m]$.

Intuition behind candidate with partially-public inputs. Starting from a cubic multilinear candidate $g(\mathbf{x}, \mathbf{y}, \mathbf{z})$ where all inputs are private, and the first input $\mathbf{x}$ is from a distribution that can be used as LWE noises, we transform it into another function $h(\mathbf{C}, \mathbf{y}', \mathbf{z})$ where the first input can be made public. The key idea is hiding $\mathbf{x}$ in LWE samples $\mathbf{C} = (\mathbf{A}, \mathbf{A}\mathbf{s}' + \mathbf{x}) \mod p$ as the noise terms. Then computing g translates into computing another function h where $\mathbf{x}$ is replaced with $\mathbf{C}\mathbf{s} \mod p$ for $\mathbf{s} = (-\mathbf{s}'||1)$,

$$h(\mathbf{C}, \mathbf{y}' = (\mathbf{y} \otimes \mathbf{s}), \mathbf{z}) := \sum_j g(\mathbf{C}[\star, j], s_j \mathbf{y}, \mathbf{z}) = g(\mathbf{C}\mathbf{s}, \mathbf{y}, \mathbf{z}) = g(\mathbf{x}, \mathbf{y}, \mathbf{z}) \pmod{p},$$

where $\mathbf{C}[\star, j]$ is the vector containing the j'th element of all LWE samples. Now $\mathbf{C}$ is the public input of h. By providing the tensor $\mathbf{y} \otimes \mathbf{s}$ as input, the polynomial h is multilinear. For h to be secure when $\mathbf{C}$ is public, the output of g needs to be indistinguishable from a pseudo flawed-smudging distribution, say $\mathcal{D}$, even when its first input is hidden in some LWE samples,

$$\{ \, g(\mathbf{x}, \mathbf{y}, \mathbf{z}), \ \mathbf{C} = (\mathbf{A}, \mathbf{A}\mathbf{s}' + \mathbf{x}) \, \} \approx \{ \, \Delta \leftarrow \mathcal{D}, \ \mathbf{C} = (\mathbf{A}, \mathbf{A}\mathbf{s}' + \mathbf{x}) \, \} \,.$$

The family of cubic polynomials with partially-public input of [AJS18] corresponds exactly to h obtained by applying the above transformation to the degree $d = 3$ candidates $g(\mathbf{x}, \mathbf{y}, \mathbf{z}) = \sum_{i_1, i_2, i_3} c_{i_1, i_2, i_3} x_{i_1} y_{i_2} z_{i_3}$ with small inputs and coefficients described in Instantiation 1. We observe that for every fixed public input $\mathbf{C}$, the function h is quadratic in $\mathbf{y}$ and $\mathbf{z}$, but its computation over $\mathbb{Z}_p$ does not degenerate to computation over $\mathbb{Z}$, as it does trigger wrap-around modulo p due to LWE "decryption".

2.4 Our Instantiation of Polynomials for ΔRG and PFG.

We now give various instantiations of Q. Let χ be the discrete gaussian distribution with 0 mean and standard deviation n. The following candidate is proposed by [BHJ$^+$19] and [AJS18] based on the investigation of the hardness of families of expanding polynomials over the reals. For any vector $\mathbf{v}$, denote by $\mathbf{v}[i]$, the ith component of the vector.

Instantiation: 3XOR Based Candidate. Let $t = B^2 \lambda$. Sample each polynomial q_i' for $i \in [\eta]$ as follows. $q_i'(\mathbf{x}_1, \ldots, \mathbf{x}_t, \mathbf{y}_1, \ldots, \mathbf{y}_t, \mathbf{z}_1, \ldots, \mathbf{z}_t) = \Sigma_{j \in [t]} q_{i,j}'(\mathbf{x_j}, \mathbf{y_j}, \mathbf{z_j})$. Here $\mathbf{x}_j \in \chi^{d \times n}$ and $\mathbf{y}_j, \mathbf{z}_j \in \chi^n$ for $j \in [t]$. In other words, q_i' is a sum of t polynomials $q_{i,j}'$ over t disjoint set of variables.

Now we describe how to sample $q_{i,j}'$ for $j \in [\eta]$.

1. To sample $q_{i,j}'$ do the following. Sample three indices randomly and independently $i_1, i_2, i_3 \leftarrow [n]$.
2. Set $q_{i,j}'(\mathbf{x_j}, \mathbf{y_j}, \mathbf{z_j}) = \mathbf{x_j}[i_1] \cdot \mathbf{y}_j[i_2] \cdot \mathbf{z}_j[i_3]$

Remark: The candidate above was generalised to have a constant degree d in a followup. This can be found in [JLMS19]. One could also consider arithmetic versions of various boolean predicates. For example, any clause of the form $a_1 \vee a_2 \vee a_3$ can be written as $1 - (1 - a_1)(1 - a_2)(1 - a_3)$ over integers where a_1, a_2, a_3 are literals in first case and take values in $\{0, 1\}$, and thus any random satisfiable 3SAT formula can be converted to polynomials in this manner.

2.5 Pseudorandomness Assumption in Ananth-Jain-Sahai

Below we describe the actual hardness assumption needed by [AJS18], when combined with subexponentially secure LWE, bilinear maps, and 3-block-local PRGs, to imply $i\mathcal{O}$.

The AJS Assumption. This assumption states the following. There exists a polynomially bounded distribution χ over the integers, and there exists a polynomial sampler $\mathcal{Q}$ over families of multilinear degree-3 polynomials. Let $\delta_i \in \mathbb{Z}$ be output by the adversary given only the parameters $(1^\lambda, 1^n)$, such that for all $i \in [n^{1+\epsilon}]$, we have that $|\delta_i| < \lambda^c$ for some constant c. Then consider the following two distributions:

Distribution $\mathcal{D}_1$: Fix a prime modulus $\mathbf{p} = O(2^\lambda)$. Run $Q(n, \lambda^c, \epsilon)$ to obtain polynomials $(q_1, ..., q_{\lfloor n^{1+\epsilon} \rfloor})$. Sample a secret $\mathbf{s} \leftarrow \mathbb{Z}_p^\lambda$ and sample $\mathbf{a}_i \leftarrow \mathbb{Z}_p^\lambda$ for $i \in [n]$. Finally, for every $i \in [n]$, sample $e_i, y_i, z_i \leftarrow \chi$, and write $\mathbf{e} = (e_1, \ldots, e_n)$, $\mathbf{y} = (y_1, \ldots, y_n)$, $\mathbf{z} = (z_1, \ldots, z_n)$. Output:

$$\{\mathbf{a}_i, \langle \mathbf{a}_i, \mathbf{s} \rangle + e_i \bmod p\}_{i \in [n]}$$

along with

$$\{q_k, q_k(\mathbf{e}, \mathbf{y}, \mathbf{z})\}_{k \in [n^{1+\epsilon}]}$$

Distribution $\mathcal{D}_2$ is the same as $\mathcal{D}_1$, except that we consider polynomial evaluations perturbed with δ_i. The output is now

$$\{\mathbf{a}_i, \langle \mathbf{a}_i, \mathbf{s} \rangle + e_i \bmod p\}_{i \in [n]}$$

along with

$$\{q_k, q_k(\mathbf{e}, \mathbf{y}, \mathbf{z}) + \delta_k\}_{k \in [n^{1+\epsilon}]}$$

Then we require that for all subexponential-time adversary $\mathcal{A}$ it holds that:

$$\left| \Pr_{Z \xleftarrow{\$} \mathcal{D}_1} [\mathcal{A}(Z) = 1] - \Pr_{Z \xleftarrow{\$} \mathcal{D}_2} [\mathcal{A}(Z) = 1] \right| \leq 1 - 1/\lambda$$

Remark 4. For concreteness, the candidate for the sampler Q can be found in Section 2.4.

Decomposing the assumption into two parts. To help understand the assumption above, next we make the following observation. The assumption described above is sufficient to build $i\mathcal{O}$ and it turns out the assumption above is true if the following two simpler assumptions are true. This implication is one sided and indeed it may be true that one of the two assumptions below is false but the assumption above still holds. We present the assumptions below only to help the reader conceptually understand the assumption above. The first assumption called "Weak LWE with Leakage" states that given the polynomial samples, it is computationally hard to determine whether the LWE sample is chosen with the same error over which the polynomials are evaluated or a completely independently chosen error.

Explaining the AJS Assumption, Part 1. Weak LWE with leakage. This assumption states that there exists a polynomially bounded distribution χ over the integers, and there exists a polynomial sampler $\mathcal{Q}$ over families of multilinear degree-3 polynomials such that the following two distributions are weakly indistinguishable (specified later).

Distribution $\mathcal{D}_1$: Fix a prime modulus $\mathbf{p} = O(2^\lambda)$. Run $Q(n, \lambda^c, \epsilon)$ to obtain polynomials $(q_1, ..., q_{\lfloor n^{1+\epsilon} \rfloor})$ for some constant $c > 0$. Sample a secret $\mathbf{s} \leftarrow \mathbb{Z}_p^\lambda$ and sample $\mathbf{a}_i \leftarrow \mathbb{Z}_p^\lambda$ for $i \in [n]$. Finally, for every $i \in [n]$, sample $e_i, y_i, z_i \leftarrow \chi$, and write $\mathbf{e} = (e_1, \ldots, e_n)$, $\mathbf{y} = (y_1, \ldots, y_n)$, $\mathbf{z} = (z_1, \ldots, z_n)$. Output:

$$\{\mathbf{a}_i, \langle \mathbf{a}_i, \mathbf{s} \rangle + e_i \bmod p\}_{i \in [n]}$$

along with

$$\{q_k, q_k(\mathbf{e}, \mathbf{y}, \mathbf{z})\}_{k \in [n^{1+\epsilon}]}$$

Distribution $\mathcal{D}_2$ is the same as $\mathcal{D}_1$, except that we additionally sample $e'_j \leftarrow \chi$ for $i \in [n]$. The output is now

$$\{\mathbf{a}_i, \langle \mathbf{a}_i, \mathbf{s} \rangle + e'_i \bmod p\}_{i \in [n]}$$

along with

$$\{q_k, q_k(\mathbf{e}, \mathbf{y}, \mathbf{z})\}_{k \in [n^{1+\epsilon}]}$$

Then it holds that for any adversary $\mathcal{A}$ of subexponential size:

$$\left| \Pr_{Z \xleftarrow{\$} \mathcal{D}_1} [\mathcal{A}(Z) = 1] - \Pr_{Z \xleftarrow{\$} \mathcal{D}_2} [\mathcal{A}(Z) = 1] \right| \leq 1/\lambda$$

We can think of the polynomials $q_k(\mathbf{e}, \mathbf{y}, \mathbf{z})$ as "leaking" some information about the LWE errors e_i. The assumption above states that such leakage provides only a limited advantage to the adversary. Critically, the fact that there are $n^2 > n^{1+\epsilon}$ quadratic monomials involving just $\mathbf{y}$ and $\mathbf{z}$ above, which are not used in the LWE samples at all, is crucial to avoiding linearization attacks over $\mathbb{Z}_p$ in the spirit of Arora-Ge [AG11]. For more discussion of the security of the above assumption in the context of $D = 3$, see [BHJ$^+$19].

The second assumption deals only with expanding degree-3 polynomials over the reals, and requires that these polynomials are weakly perturbation resilient.

Explaining the AJS Assumption, Part 2. Weak Perturbation-Resilience. This assumption states that for the same distribution of polynomials and inputs as above the following distributions are weakly indistinguishable. Let $\delta_i \in \mathbb{Z}$ be output by the adversary given only the parameters $(1^\lambda, 1^n)$, such that for all $i \in [n^{1+\epsilon}]$, we have that $|\delta_i| < \lambda^c$ for some constant c. Consider the following two distributions:

Distribution $\mathcal{D}_1$ consists of the evaluated polynomial samples. That is, we output:

$$\{q_k, q_k(\mathbf{e}, \mathbf{y}, \mathbf{z})\}_{k \in [n^{1+\epsilon}]}$$

Distribution $\mathcal{D}_2$ consists of the evaluated polynomial samples with added perturbations δ_i for $i \in [n^{1+\epsilon}]$. That is, we output:

$$\{q_k, q_k(\mathbf{e}, \mathbf{y}, \mathbf{z}) + \delta_k\}_{k \in [n^{1+\epsilon}]}$$

Then it holds that for any adversary $\mathcal{A}$ of subexponential size:

$$\left| \Pr_{Z \xleftarrow{\$} \mathcal{D}_1} [\mathcal{A}(Z) = 1] - \Pr_{Z \xleftarrow{\$} \mathcal{D}_2} [\mathcal{A}(Z) = 1] \right| \leq 1 - 3/\lambda$$

2.6 Known Cryptanalysis

Now, we discuss various preliminary cryptanalysis attempts made on these candidates. These attacks can be categorised in the following categories:

Linearisation Attacks: The system of degree-3 polynomials described above can be converted to a degree-2 system over $\mathbb{Z}_\mathbf{p}$ by performing back substitution of e_i, from the LWE sample $(a_i, \langle a_i, s \rangle + e_i \mod \mathbf{p})$. However, the resulting system has about $\Omega(n)$ variables $\mathbf{y} \otimes \mathbf{s}$ and $\mathbf{z}$, but only about $n^{1+\epsilon}$ equations. Thus, all known linearization attack fail. This was considered in the work of [BHJ$^+$19].

Sum-of-Squares Attacks: [BHJ$^+$19] systematically studies SDP attacks on such system and they gave an evidence why the assumptions above instantiated using degree-2 polynomials over reals is unlikely to be true. However, they also conjecture that for degree-3 and higher, these systems exhibit SoS lower bounds (at least, the lower bounds are known to hold in the case when inputs are chosen from $\{-1, 1\}$ [Gri01, Sch08]). The lower bounds hold when number of equations $m \leq n^{d/2}$ for a general degree $d \geq 3$. Thus for our case when $m = n^{1+\epsilon}$ for any $\epsilon > 0$, the SoS algorithm is unlikely to attack such systems in polynomial time. Please refer [BHJ$^+$19] for further details.

Gradient Descent: We implemented gradient descent to cryptanalyze all our candidates. It seems like given the signs of the planted inputs, gradient descent was able to recover the planted inputs in most cases. For degree-2 candidates, gradient descent was able to recover the planted inputs even with random starting points (even with no information on the signs). For degree-3 and higher, our implementation of gradient descent did not yield any attack starting from random signs. This matches our intuition developed in SoS literature, since the lower bounds hold when inputs are sampled from $\{+1, -1\}$ (thus implying finding signs is hard).

3 Technical Overview of Ananth-Jain-Sahai 18

We now begin with a very high-level overview of our techniques in [AJS18].

The story so far. Prior work, culminating in the most recent works of [AS17, Lin17, LT17] showed us that the powerful primitive of indistinguishability obfuscation can be based on trilinear maps and (sub-exponential) 3-block-local pseudorandom generators. Importantly for us, these works also (implicitly) demonstrate that in order to achieve indistinguishability obfuscation, it suffices to construct (sub-exponentially secure) secret-key sublinear FE for cubic polynomials, satisfying semi-functional security. Unfortunately, these prior approaches necessarily relied on multilinear maps with degree at least 3 to build such a cubic FE scheme.

That is because intuitively such a cubic FE scheme guarantees a way to evaluate a cubic polynomial on encrypted inputs without revealing any information about the input except the evaluation of the polynomial. In other words, such

a scheme provides a way to output the decryption of a degree-3 polynomial evaluated "homomorphically" on encoded inputs. However, we seek to accomplish this without the use of degree-3 maps.

Since we seek to operate homomorphically on encoded values, a natural starting idea is to use fully homomorphic encryption (for concreteness and simplicity, in this paper we rely on the GSW fully homomorphic encryption scheme [GSW13]) with polynomially bounded error in order to perform cubic evaluations on encrypted inputs. The main challenge, however, is to reveal the output of cubic evaluation without compromising security.

Initial approach. Our first observation is that computing the inner product $\langle \mathsf{GSW.sk}, \mathsf{GSW.CT} \rangle$ of a GSW secret key with a GSW ciphertext encrypting message M, outputs $(M \cdot \lfloor q/2 \rfloor + e)$ where the LWE modulus is q and e is a small error. With the assistance of a bilinear map, this inner product can be carried out via pairings, such that the output $(M \cdot \lfloor q/2 \rfloor + e)$ appears as an exponent in the target group. Next, one can hope to test whether the message M is zero by computing a discrete logarithm by brute-force checking all possible values, provided the output range is polynomial, which would happen if $M = 0$.

A reader familiar with GSW will observe that this approach already runs into major hurdles. The first problem is that brute-force computing the message M also reveals the error e to a potential adversary, which is problematic when we try to invoke the semantic security of GSW. In fact, recent work shows how knowledge of such error can be used to build devastating attacks [Agr17b]. We will crucially deal with this issue, but before we tackle this, let us first consider how we can force the adversary to obtain only inner products $\langle \mathsf{GSW.sk}, \mathsf{GSW.CT} \rangle$ where the messages correspond to cubic computations that the adversary is allowed to obtain.

3-Restricted FE. To accomplish this, we first define a *restricted* version of functional encryption (FE) – which allows for the computation of multilinear cubic polynomials of three inputs, where one remains unencoded and is called the public component and the other two are encoded; these are the private components. In other words, our restricted FE is a *partially hiding* FE, or PHFE for short. The input to the encryption algorithm is split into three parts $\mathbf{x}, \mathbf{y}$, and $\mathbf{z}$, where $\mathbf{x}$ is not hidden by the encryption, but $\mathbf{y}$ and $\mathbf{z}$ are kept hidden.

One of our key technical contributions is to achieve a new way of (indistinguishably) enforcing the output of such a 3-restricted FE scheme, despite the fact that one of the encodings is publicly known to the adversary. We use these techniques to achieve security for this 3-restricted variant of FE relying solely on asymmetric bilinear maps. While we only need the resulting 3-restricted FE to be sublinear, our construction in fact achieves compactness, where the size of encoding is only linear in the input length.

Constructing Three-Restricted FE. Before getting to 3 restricted FE, let's first recap how secret key quadratic functional encryption schemes [AS17, Lin17] work at a high level. Let's say that the encryptor wants to encrypt $\mathbf{y}, \mathbf{z} \in \mathbb{Z}_{\mathbf{p}}^n$. The

22

master secret key consists of two secret random vectors $\beta, \gamma \in \mathbb{Z}_{\mathbf{p}}^n$ that are used for enforcement of computations done on $\mathbf{y}$ and $\mathbf{z}$ respectively. The idea is that the encryptor encodes $\mathbf{y}$ and β using some randomness r, and similarly encodes $\mathbf{z}$ and γ together as well. These encodings are created using bilinear maps in one of the two base groups. These encodings are constructed so that the decryptor can compute an encoding of $[g(\mathbf{y}, \mathbf{z}) - rg(\beta, \gamma)]_t$ in the target group for *any* quadratic function g. The function key for the given function f is constructed in such a manner that it allows the decryptor to compute the encoding $[rf(\beta, \gamma)]_t$ in the target group. Thus the output $[f(\mathbf{y}, \mathbf{z})]_t$ can be recovered in the exponent by computing the sum of $[rf(\beta, \gamma)]_t$ and $[f(\mathbf{y}, \mathbf{z}) - rf(\beta, \gamma)]_t$ in the exponent. As long as $f(\mathbf{y}, \mathbf{z})$ is polynomially small, this value can then be recovered efficiently.

Clearly the idea above only works for degree-2 computations, if we use bilinear maps. However, we build upon this idea nevertheless to construct a 3-restricted FE scheme. Recall, in a 3-restricted FE one wants to encrypt three vectors $\mathbf{x}, \mathbf{y}, \mathbf{z} \in \mathbb{Z}_{\mathbf{p}}^n$. While $\mathbf{y}$ and $\mathbf{z}$ are required to be hidden, $\mathbf{x}$ is not required to be hidden.

Now, in addition to $\beta, \gamma \in \mathbb{Z}_{\mathbf{p}}^n$ in case of a quadratic FE, another vector $\alpha \in \mathbb{Z}_{\mathbf{p}}^n$ is also sampled that is used to enforce the correctness of the $\mathbf{x}$ part of the computation. As before, given the ciphertext one can compute $[\mathbf{y}[j]\mathbf{z}[k] - r\beta[j]\gamma[k]]_t$ for $j, k \in [n]$. But this is clearly not enough, as these encodings do not involve $\mathbf{x}$ in any way. Thus, in addition, an encoding of $r(\mathbf{x}[i] - \alpha[i])$ is also given in the ciphertext for $i \in [n]$. Inside the function key, there are corresponding encodings of $\beta[j]\gamma[k]$ for $j, k \in [n]$ which the decryptor can pair with encoding of $r(\mathbf{x}[i] - \alpha[i])$ to form the encoding $[r(\mathbf{x}[i] - \alpha[i])\beta[j]\gamma[k]]_t$ in the target group.

Now observe that,

$$\mathbf{x}[i] \cdot \big(\mathbf{y}[j]\mathbf{z}[k] - r\beta[j]\gamma[k]\big) + r(\mathbf{x}[i] - \alpha[i]) \cdot \beta[j]\gamma[k]$$
$$= \mathbf{x}[i]\mathbf{y}[j]\mathbf{z}[k] - r\alpha[i]\beta[j]\gamma[k]$$

Above, since $\mathbf{x}[i]$ is public, the decryptor can herself take $(\mathbf{y}[j]\mathbf{z}[k] - r\beta[j]\gamma[k])$, which she already has, and multiply it with $\mathbf{x}[i]$ in the exponent. This allows her to compute encoding of $[\mathbf{x}[i]\mathbf{y}[j]\mathbf{z}[k] - r\alpha[i]\beta[j]\gamma[k]]_t$. Combining these encodings appropriately, she can obtain $[g(\mathbf{x}, \mathbf{y}, \mathbf{z}) - rg(\alpha, \beta, \gamma)]_t$ for any degree-3 multilinear function g. Given the function key for f and the ciphertext, one can compute $[rf(\alpha, \beta, \gamma)]_t$ which can be used to unmask the output. This is because the ciphertext contains an encoding of r in one of the base groups and the function key contains an encoding of $f(\alpha, \beta, \gamma)$ in the other group and pairing them results in $[rf(\alpha, \beta, \gamma)]_t$.

In full version [AJS18], we provide details of our 3-restricted FE; specifically, we define a notion of semi-functional security [AS17] (variant of function-hiding) associated with a three-restricted FE scheme. Once we have such a restricted FE, making the leap to cubic FE would require us to also keep the public encoding hidden. Therefore, it is not clear whether we have achieved anything meaningful yet.

Applying Three-Restricted FE. One way that we can hope to protect or hide the input that goes into the public component of the 3-restricted FE, is to let this

component itself be a GSW-based fully homomorphic encryption of the input. We can then rely on 3-restricted FE to homomorphically evaluate the cubic function itself and obtain a GSW encryption of the output of cubic evaluation. Note, however, that releasing such a GSW encryption by itself is useless, because it does not allow even an honest evaluator to recover the output of cubic evaluation.

At this point, let us go back to the initial approach described at the beginning of this section. Notice that instead of relying on 3-restricted FE to *only* homomorphically evaluate the cubic function itself, we can also perform a GSW decryption via 3-restricted FE. The secret key for GSW decryption can be embedded as input into one of the private components of the 3-restricted FE. We show how this can be carefully done via degree three operations only, to obtain output the GSW plaintext with some added error, that is, we obtain $\mathsf{out} = \mu\lfloor\frac{q}{2}\rfloor + e$. Our actual method of bootstrapping three-restricted FE to sublinear FE for cubic polynomials involves additional subtleties, and in particular, we define and construct what we call *tempered cubic encodings* that serve as a useful abstraction in this process. We now further discuss one of the main technical issues that arises in this process.

Because the error e is sampled from a (bounded) polynomial-sized domain, it is possible to iterate, in polynomial time, over all possible values of out corresponding to $\mu = 0$ and $\mu = 1$, and therefore recover μ. Unfortunately, this process also reveals the error e, which can be devastating as we noted before.

Preventing the revelation of error terms. To prevent this issue, we will reveal the value $\mathsf{out} = \mu\lfloor\frac{q}{2}\rfloor + e$ but with some added noise, so as to hide the error e via noise flooding. Unfortunately, this idea still suffers from two major drawbacks:

- How should we generate such noise? A natural idea is to rely a pseudorandom generator that can be computed via quadratic operations only. However, this is exactly the reason why previous approaches from the literature could not rely on bilinear maps – in fact, the recent works of [LV17, BBKK17] showed that such PRGs are quite difficult to construct. To overcome this problem, we introduce and rely on a very weak variant of a pseudorandom object, that instead of guaranteeing pseudorandomness, only guarantees perturbation resilience. Furthermore, we will implement this object with degree-3 polynomials. We will soon explain this object in more detail.
- For an honest evaluator to recover μ by iterating over all possible values of $\mathsf{out} = \mu\lfloor\frac{q}{2}\rfloor + e$, we crucially require the added noise be sampled from a polynomial-sized domain. But such noise appears to be insufficient for security, in particular, an adversary would have advantage at least $\frac{1}{\mathrm{poly}(\lambda)}$ in distinguishing a message with added noise from a message without noise. Another key technical contribution of our work is to find a way to *amplify security*, via tools inspired by the dense model theorem. In the next two bullets, we describe these ideas in additional detail.

The challenge of constructing degree-3 pseudorandomness: a barrier at degree 2. As we've outlined above, we need a way to create pseudorandomness to

(at least partially) hide noise values. The most straightforward way to do this would be to build a degree-2 pseudorandom generator (PRG) whose output is indistinguishable from some nice m-dimensional distribution, like a discrete gaussian. Intuitively, if such a degree-2 object existed, a bilinear map would be sufficient to implement it. However, the works of [BBKK17, LV17] showed that there are fundamental barriers to constructing such PRGs due to attacks arising from the Sum of Squares paradigm. Because we will propose a direction to overcome this barrier, we now review how these attacks work at a high level.

For simplicity, let's restrict our attention to polynomials where every monomial is of degree exactly 2. We can represent any such polynomial p as a symmetric n-by-n matrix P, where $P_{i,j} = P_{j,i}$ is equal to half the coefficient of the monomial $x_i x_j$ if $i \neq j$, and $P_{i,i}$ is equal to the coefficient of the monomial x_i^2. Then we observe that $p(x) = x^\top P x$. Suppose, then, we have a candidate PRG consisting of m degree-2 polynomials that we represent by matrices $M_1, \ldots, M_m$. Thus, to sample from this PRG, we sample a seed vector x from a bounded-norm distribution, and obtain the outputs $y_i = x^\top M_i x$. The goal of an attack would be to distinguish such outputs from a set of independent random values $r_1, \ldots, r_m$, say from a discrete gaussian distribution centered around zero.

The works of [BBKK17, LV17] suggest the following attack approach: Suppose we receive values $z_1, \ldots, z_m$. Then we construct the matrix

$$M = \sum_{i=1}^{m} z_i M_i$$

Observe now, that if $z_i = y_i$ corresponding to some seed vector x, then we have:

$$x^\top M x = \sum_{i=1}^{m} y_i x^\top M_i x = \sum_{i=1}^{m} y_i^2$$

Intuitively, because the above sum is a sum of squares, this will be a quite large positive value, showing that there exists x of bounded norm such that $x^\top M x$ can be quite large.

However, if the $z_i = r_i$, then the entries of the matrix M arise from a "random walk," and thus intuitively, the matrix M should behave a lot like a random matrix. However a random matrix has bounded eigenvalues, and thus we expect that there should not exist any x of bounded norm such that $x^\top M x$ is large. Indeed, this intuition can be made formal and gives rise to actual attacks on many degree-2 PRGs [BBKK17, LV17]. The attack above was generalized further in a followup work to this paper [BHJ+19], showing that several families of degree-2 pseudorandom objects cannot exist. While there are still potential caveats to known degree-2 attacks, we propose a different, more conservative, way forward:

Perturbation-Resilient Generators (ΔRG). We observe that even though the most natural way to "drown out" the GSW error term above is by adding some nice noise distribution, all we actually need is something we will call a perturbation-resilient generator (ΔRG): Informally speaking, we want that for every polynomial bound

25

$B(\lambda)$, there should exist a low-degree[10] ΔRG using polynomially bounded seeds and coefficients, such that for any perturbation vector $a \in [-B, B]^m$, it should be true that all efficient adversaries must fail to distinguish between the distributions $\Delta RG(x)$ and $(\Delta RG(x) + a)$ with probability at least $1/poly(\lambda)$, which is a fixed inverse polynomial in the security parameter. We stress again that we are not seeking a ΔRG with standard negligible security, but only some low level of security. Indeed, even if an efficient adversary could distinguish between $\Delta RG(x)$ and $(\Delta RG(x) + a)$ with probability $1 - 1/poly(\lambda)$, but still fail to distinguish on at least $1/poly(\lambda)$ probability mass, our approach will succeed due to amplification (see below).

Crucially, instead of requiring the ΔRG to be computable via polynomials of degree two, we define a notion of ΔRG implementable by degree three polynomials via our notion of 3-restricted FE.

The seed for a ΔRG consists of one public and two private components, and perturbation-resilience is required even when the adversary has access to the public component of the seed. Furthermore, the use of cubic (as opposed to quadratic) polynomials gives reason to hope that our ΔRGs do not suffer from inversion attacks and achieve the weak form of security described above. Further in-depth research is certainly needed to explore our new assumptions. Indeed, we see our work as strongly motivating the systematic exploration of the limits of various types of low degree pseudorandom objects over $\mathbb{Z}$ using the Sum of Squares paradigm and beyond. Indeed, our work reveals a fascinating connection between achieving $i\mathcal{O}$ and studying distributions of expanding low-degree polynomials over the reals that are hard to solve. We refer the reader to [BHJ+19] for further discussion on this topic.

Implementing Degree-3 ΔRGs. Having constructed a three-restricted FE scheme, we now describe how to implement the degree-3 ΔRG as described above. Let $\mathbf{e} = (e_1, \ldots, e_n)$, $\mathbf{y} = (y_1, \ldots, y_n)$ and $\mathbf{z} = (z_1, ..., z_n)$ and we want to compute degree three polynomials of the form $q_\ell(\mathbf{e}, \mathbf{y}, \mathbf{z}) = \Sigma_{\mathbf{I}=(i,j,k)} c_{\mathbf{I}} \cdot e_i \cdot y_j \cdot z_k$ where $\ell \in [\eta]$ is the stretch. Here all variables and coefficients are polynomially bounded in absolute value.

At first glance, one could think to could encrypt $\mathbf{e}$ in the public component and $\mathbf{y}, \mathbf{z}$ in the private component of the three restricted FE scheme. Then, one could issue function keys for polynomials q_ℓ for $\ell \in [\eta]$. However, such a scheme would essentially yield a degree 2 system of polynomials in $\mathbf{y}$ and $\mathbf{z}$ as $\mathbf{e}$ is public, and not provide any additional security beyond using degree-2 polynomials. In order to fix this issue, we take a different approach.

Encrypting $\mathbf{e}$ as an LWE-style error. Instead, we sample a secret $\mathbf{s} \in \mathbb{Z}_{\mathbf{p}}^{\mathsf{d}}$ where d is some polynomial in the security parameter. We also sample vectors $\mathbf{a_i} \leftarrow \mathbb{Z}_{\mathbf{p}}^{\mathsf{d}}$ for $i \in [n]$. Then we compute $r_i = \langle \mathbf{a_i}, \mathbf{s} \rangle + e_i$. Let $\mathbf{w_i} = (\mathbf{a_i}, r_i)$ for $i \in [n]$. Thus

---

[10]   In an earlier version of this paper, this overview focused on constructing degree-2 ΔRGs. However, as we describe now, our technical approach is more general, and we describe it in greater generality here.

we have encrypted $\mathbf{e}$ using the secret $\mathbf{s}$. Now to implement degree-3 randomness generator we consider the polynomial:

$$q_\ell(\mathbf{e}, \mathbf{y}, \mathbf{z}) = \Sigma_{\mathbf{I}=(i,j,k)} c_{\mathbf{I}} \cdot e_i \cdot y_j \cdot z_k$$

This polynomial can be re-written as:

$$q_\ell(\mathbf{e}, \mathbf{y}, \mathbf{z}) = \Sigma_{\mathbf{I}=(i,j,k)} c_{\mathbf{I}} \cdot (r_i - \langle \mathbf{a_i}, \mathbf{s} \rangle) \cdot y_j \cdot z_k$$

Now suppose in the private component that contained $\mathbf{y}$, we also put $\mathbf{y} \otimes \mathbf{s}$ (where $\otimes$ denotes the tensor operation). Then observe that if $\mathbf{w}_i$ for $i \in [n]$ are all public values, then the entire polynomial can now be computed using a three-restricted FE scheme.

For this approach to be secure, intuitively we want that $\mathbf{e}$ is sampled from an "error" distribution with respect to which the LWE assumption holds. (For simplicity, we can think of $\mathbf{y}$ and $\mathbf{z}$ also being sampled from such a distribution.) The security of our ΔRG would then rely on a variant of the LWE assumption. Experience teaches that one should be cautious when considering the security of variants of LWE, and our case is no exception. This variant was studied in a follow-up work of [BHJ+19], where several unsuccessful attacks were considered. We briefly review one of these now. The most common source of devastating attacks to LWE variants is linearization. However, a key barrier to such attacks in our setting is the fact that the LWE-based public values $\mathbf{w}_i$ contain *no information whatsoever* about $\mathbf{y}$ and $\mathbf{z}$. Thus, over $\mathbb{Z}_{\mathbf{p}}$, we would obtain a set of roughly $n^{1+\epsilon}$ quadratic equations in $\mathbf{y} \otimes \mathbf{s}$ and $\mathbf{z}$, but crucially with *large* coefficients in $\mathbb{Z}_{\mathbf{p}}$. These large coefficients would arise from the fact that r_i and $\mathbf{a}_i$ are large values. Such systems, called MQ systems, have been widely studied cryptanalytically and are widely believed to be hard to solve [Wol02, KS99] in general. We again refer the reader to [BHJ+19] for further discussion. Specific candidates for the degree-3 polynomials q_ℓ above, inspired by the hardness of RANDOM 3-SAT and suggested by [BHJ+19], are also given in Section 2.

Security Amplification. Crucially, we want allow an adversary to have a very large distinguishing advantage when attempting to distinguish between $\Delta RG(x)$ and $(\Delta RG(x) + a)$, since this is a new assumption. For simplicity for this technical overview, we will assume that the ΔRG we introduce above is $\frac{1}{\lambda}$-secure. (More generally, we can tolerate any fixed inverse polynomial in the security parameter.)

Using ideas already discussed above, it is possible to show (as we do in our technical sections) that relying on $\frac{1}{\lambda}$-secure ΔRG in the approach outlined above, allows us to achieve a "weak" form of sublinear FE (sFE), that only bounds adversarial advantage by $\frac{1}{\lambda}$. Unfortunately, such an FE scheme it not known to yield $i\mathcal{O}$, and for our approach to succeed, we must find a way to amplify security of sublinear FE.

How should we amplify security? An initial idea is to implement a direct-product type theorem for functional encryption. However, a simple XOR trick does not suffice: since we are trying to amplify security of a complex primitive like FE while retaining correctness, we will additionally need to rely on a special

kind of secure computation. More precisely, we will use (subexponentially secure) n-out-of-n threshold fully homomorphic encryption (TFHE [MW16, BGG$^+$18]), that is known to exist based on LWE [Reg05]. Recall that such a threshold (public key) fully homomorphic encryption scheme allows to encrypt a ciphertext in such a way that all secret key holders can *partially* decrypt the ciphertext, and then can recover the plaintext by combining these partial decryptions. However, any coalition of secret key holders of size at most $n - 1$ learns no information about the message.

A simplified overview of our scheme, that makes use of $t = \lambda^2$ weak sublinear FEs, is as follows:

- The setup algorithm outputs the master secret keys msk_i for all weak sublinear FEs.
- In order to generate the encryption of a plaintext M, generate a public key TFHE.pk and t fresh secret keys TFHE.sk_i for a threshold FHE, and encrypt M using the public key for threshold FHE to obtain ciphertext TFHE.ct. Additionally, for all i, encrypt (TFHE.ct, TFHE.sk_i) using the master secret key msk_i for the ith weak sublinear FE.
- To generate a function secret key for circuit C, generate t function secret keys for the sFEs, each of which computes the output of the ith TFHE partial decryption of the result of homomorphic evaluation of the circuit C on TFHE.ct.
- Finally, to evaluate a functional secret key for circuit C on a ciphertext, combine the results of the TFHE threshold decryptions obtained via the t outputs of sFE evaluation of the t function secret keys.

The correctness of our scheme follows immediately from the correctness properties of the TFHE scheme. *Intuitively*, security seems to hold because of the following argument. Upon combining λ^2 independent, random instances of the weak sFE, with overwhelming probability, at least one must remain secure. As long as a single instance remains secure, the corresponding secret key for TFHE will remain hidden from the adversary. Now, TFHE guarantees semantic security against any adversary that fails to obtain even one secret key, and as a result, the resulting FE scheme should be secure. While this intuition sounds deceptively simple, many of these intuitive leaps assume information-theoretic security. Thus, this template evades a formal proof in the computational setting, and we must work harder to obtain our proof of security, as we now sketch.

From a cryptographic point of view, one of the early hurdles when trying to obtain such a proof is the following. A reduction must rely on an adversary that breaks security of the final FE scheme with *any* noticeable probability, in order to break $\frac{1}{\lambda}$ security of one of the λ^2 "weak" FEs. However, the reduction does not know which of the λ^2 repetitions is secure, and therefore does not directly know where to embed an external challenge. To deal with this, we rely on the concept of a *hardcore measure* [Imp95, MT10]. Roughly speaking, we obtain measures of probability mass roughly $\frac{1}{\lambda}$ over the randomness of the sFE schemes, such that no efficient adversary can break the security of the sFE scheme even with some inverse subexponential probability.

However, unfortunately these hardcore measures can depend on other parameters in our system, such as the TFHE public key. And unfortunately, this dependence is via extreme inefficiency; the hardcore measure is not efficiently sampleable. This means that, for example, the hardcore measure could in principle contain information about the TFHE master secret key. If this information is leaked to the adversary, this would destroy the security of our scheme.

We overcome this issue through the following idea, which can be made formal via the work on simulating auxiliary input [JP14, CCL18]. Because the hardcore measure has reasonable probability mass $\frac{1}{\lambda}$, it cannot *verifiably* contain useful information to the adversary. For example, even if the hardcore distribution revealed the first few bits of the TFHE master secret key, the adversary could not *know* for sure that these bits were in fact the correct bits. Indeed, we use the works of [JP14, CCL18] to make this idea precise, and show that the hardcore measures can be simulated in a way that fools all efficient adversaries, with a simulation that runs in subexponential time.

Finally, using complexity leveraging, we can set the security of the TFHE scheme to be such that its security holds against adversaries whose running time exceeds this simulation. Thus, for example, even if the original hardcore measure was revealing partial information about the TFHE master secret key, we show that we can give the adversary access to a simulated hardcore measure that provably does not reveal any useful information about the TFHE master secret key, and the adversary can't tell the difference!

In this way, we accomplish security amplification for sFE, which allows us to achieve $i\mathcal{O}$ for general circuits when combined with previous work [AS17, LT17]. Along the way, our amplification technique also shows that we can weaken the security requirement on the relatively new notion of a 3-block-local PRG due to [LT17], in a way that still allows us to achieve $i\mathcal{O}$. Our amplification result can be stated as the following theorem.

Theorem 5. *Assuming there exists a constant $c > 0$ and there exists:*

- $(2^{\lambda^c}, \mathsf{adv} = 1 - 1/\lambda)-$*secure sublinear semi-functional FE scheme for $\mathcal{C}_{n',s'}$.*
- $(2^{\lambda^c}, 2^{-\lambda^c})-$*secure threshold homomorphic encryption scheme.*
- $(2^{\lambda^c}, 2^{-\lambda^c})-$*secure PRFs in NC^1.*
- $(2^{\lambda^c}, 2^{-\lambda^c})-$*secure statistically binding commitments.*

There exists a sublinear secret key FE scheme for circuit class $\mathcal{C}_{n,s}$ with $(2^{\lambda^{c'}}, 2^{-\lambda^{c'}})$ security for some constant $c' > 0$.

Combining these ideas, we obtain the following result.

Theorem 6. *Assuming*

- *LWE secure against subexponential sized circuits.*
- *Secure Three restricted FE scheme.*
- *PRGs with*
  - *Stretch of $k^{1+\epsilon}$ (length of input being k bits) for some constant $\epsilon > 0$.*
  - *Block locality three.*

- *Security with* negl *distinguishing gap against adversaries of subexponential size.*
  - *Perturbation resilient generators implementable by three restricted FE scheme with:*
    - *Stretch of $k^{1+\epsilon}$ for some $\epsilon > 0$.*
    - *Security with distinguishing gap $1 - 1/\lambda$ against adversaries of subexponential size.*

there exists a secure iO scheme for P/poly.

In a follow-up to our work [JLMS19] showed a construction of a d-restricted FE scheme for any constant $d \geq 3$ from SXDH over bilinear maps.

Theorem 7 ([JS18, LM18, JLMS19]). *Assuming SXDH over bilinear maps, there exists a construction of a three-restricted FE scheme.*

Thus, in full generality we can prove the following result.

Theorem 8. *Let* $\mathsf{adv}_1, \mathsf{adv}_2$ *be two distinguishing gaps such that* $\mathsf{adv}_1 + \mathsf{adv}_2 \leq 1 - 1/p(\lambda)$ *for any fixed polynomial $p(\lambda) > 1$. Then assuming,*

- *LWE secure against adversaries of subexponential size.*
- *SXDH secure against adversaries of subexponential size.*
- *PRGs with*
  - *Stretch of $k^{1+\epsilon}$ (length of input being k bits) for some constant $\epsilon > 0$.*
  - *Block locality three.*
  - *Security with distinguishing gap bounded by adv_1 against adversaries of subexponential size.*
- *Perturbation resilient generators implementable by three restricted FE scheme with:*
  - *Stretch of $k^{1+\epsilon}$ for some $\epsilon > 0$.*
  - *Security with distinguishing gap adv_2 against adversaries of subexponential size.*

there exists a secure iO scheme for P/poly.

3.1 Reader's Guide

In the technical overview and the introduction, we have already described our notions of three restricted FE scheme and perturbation resilient generator (ΔRG). In the sequel, for clarity, we will denote by 3ΔRG a ΔRG that is implementable by three restricted FE. Below we give a high level description of various terms used above that we have not already discussed.

Tempered Cubic Encoding: Tempered cubic encoding is a natural abstraction encapsulating a 3ΔRG and cubic homomorphic evaluation. This framework is compatible with our notion of a three restricted FE scheme and is used to build Functional Encryption for cubic polynomials.

Semi-Functional FE for cubic polynomials. A semi-functional FE scheme for cubic polynomials (FE_3 for short) is a secret key functional encryption scheme supporting evaluation for cubic polynomials where the size of the ciphertext is linear in the number of inputs. It satisfies semi-functional security: where you can hard code secret values in the function key which will be decrypted only using a single special ciphertext (known as a semi-functional ciphertext). Note that all our primitives satisfy $1 - 1/\mathrm{poly}(\lambda)$ security. They are finally amplified to construct fully secure primitives.
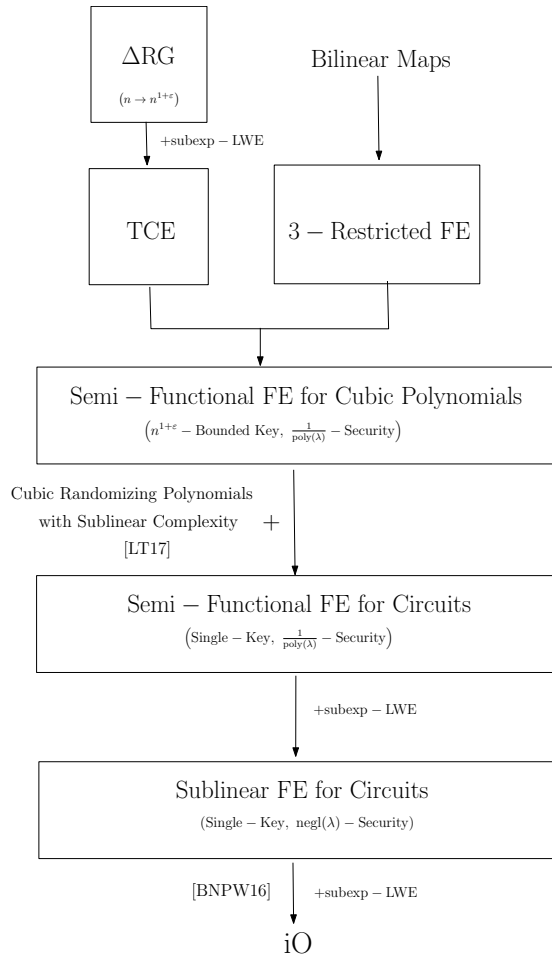
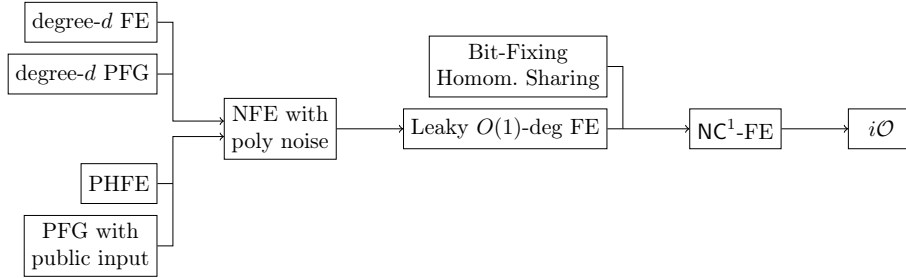**Fig. 1.** Steps involved in the construction of $i\mathcal{O}$ in [AJS18].

Fig. 2. Overview of constructions in [LM18] leading to $i\mathcal{O}$.

Semi-Functional FE for Circuits. A semi-functional FE scheme for circuits is a secret key functional encryption scheme supporting evaluation of circuits where the size of the ciphertext is sublinear in the maximum size of circuit supported. This notion also satisfies semi-functional security.

We present a diagrammatic view of construction of $i\mathcal{O}$ in Figure 3.1.

4 Technical Overview of Lin-Matt 18

We now describe techniques in [LM18] in more detail. An overview is depicted in Figure 2.

NC^1-FE from PFGs and FE that computes them. It is known that to construct $i\mathcal{O}$, it suffices to construct secret-key FE schemes for computing NC^1 circuits that have *sublinearly compact* ciphertexts of size polynomial in the security parameter λ and input length N, and sublinear in the size S of the circuits computed. Towards constructing functional encryption schemes for NC^1, we follow the same two-step approach as previous works [Lin16b, LV16, Lin17, AS17]: They showed that the task of constructing NC^1-FE can be reduced to the task of constructing FE for computing NC^0 functions, i.e., constant-degree constant-locality polynomials, by converting any NC^1 function into a NC^0 function using randomized encoding and a low locality PRG. In this work, we develop a new technique for constructing constant-degree FE and a new bootstrapping method to NC^1-FE that is "leakage resilient".

Basic Ideas: Constant-degree FE via HE and Noisy Linear FE. Existing compact constant-degree FE schemes [GGHZ16, AS17, Lin17] use multilinear map groups to directly compute the constant-degree polynomial in the exponent. We here explore a different natural approach, that has already appeared in the literature [GVW12, GVW15, BTVW17, GKP+13, AR17, Agr18b] and that performs the computation homomorphically over the encrypted input via an HE scheme. The output ciphertext is eventually decrypted using multilinear maps.

The rough template is as follows: Let the FE scheme encrypt an input $\mathbf{x}$ using an HE scheme and a secret vector $\mathbf{s}$ to obtain a ciphertext $\mathbf{c}$. To compute a function f on $\mathbf{x}$, the decryptor can homomorphically evaluate f on $\mathbf{c}$ and obtain a ciphertext CT_f encrypting the output $\mathbf{y} = f(\mathbf{x})$. The two challenges are

– **privacy**—how to decrypt CT_f in a secure way that reveals only $\mathbf{y}$ and hides all other information about $\mathbf{x}$, and
– **integrity**—how to enforce that only ciphertexts associated with a "legitimate" function f (ones for which secret keys have been generated) can be decrypted.

Previous works [GVW12, GKP+13, GVW15, BTVW17, AR17, Agr18b] developed novel techniques for achieving privacy and integrity, using various tools from garbled circuits, partially hiding predicate encryption, to noisy linear FE. But the resulting schemes either achieve weaker security guarantees as in Predicate Encryption [GVW15, BTVW17], or lose ciphertext compactness [GKP+13, AR17], or make use of strong primitives that are themselves hard to instantiate [GVW12, Agr18b]. Building upon their techniques, we propose new ones toward solving the challenges.

Observe that the decryption of most HE schemes, such as [BV11, BGV12] based on LWE, involves $i)$ a linear operation, $L_{\mathrm{dec}}(\mathsf{CT}_f, \mathbf{s})$ (e.g., $\langle \mathsf{CT}_f, \mathbf{s} \rangle$), which produces an *approximate* output, $\mathbf{y} + 2\mathbf{e}$, perturbed by a small noise vector $\mathbf{e}$, referred to as "half-decrypt", $ii)$ followed by a threshold function (complex, in NC^1) to remove the noise. Privacy entails that we must hide the secret $\mathbf{s}$ and the noise $\mathbf{e}$. Hiding the secret is relatively easy as we have FE schemes for computing a linear function, here L, over a secret, here $\mathbf{s}$, from various assumptions (e.g., DDH, LWE, Paillier). However, the output of the linear FE would be the approximate output $\mathbf{y} + 2\mathbf{e}$, and the noise $\mathbf{e}$ is *sensitive*, revealing information about the input $\mathbf{x}$, the noises used for generating the original ciphertext $\mathbf{c}$ encrypting $\mathbf{x}$, and (indirectly) the secret $\mathbf{s}$. On the other hand, removing the noise $\mathbf{e}$ requires a high-degree computation (such as mod2). The works of [AR17, Agr18b] propose to hide $\mathbf{e}$ using another bigger smudging noise—compute instead the approximate output $\mathbf{y} + 2\mathbf{e} + 2\mathbf{Y}$ further shifted by a large noise $\mathbf{Y}$ that hides $\mathbf{e}$. Agrawal [Agr18b] further encapsulated the task to be done in a primitive called *noisy linear FE*, which performs a linear computation, here the half-decrypt, and adds a *fresh* noise to the decrypted output of every pair of ciphertext and secret key. Let us now delve deeper into noisy linear FE.

4.1 Noisy Linear Functional Encryption

Noisy secret-key FE schemes have the same syntax as regular secret-key FE schemes, but decrypting a ciphertext nct of $\mathbf{v}$ with a secret key nsk_L for a linear function L yields a perturbed output $L(\mathbf{v}) + \mathbf{Y}$ (over $\mathbb{Z}_p$ for some modulus p), where the noise $\mathbf{Y}$ is distributed indistinguishably to a distribution η — we call such a scheme a η-noisy linear FE. We further only require weak correctness in the sense that decryption only needs to succeed if all coordinates of $L(\mathbf{v})$ lie in a polynomially sized range, and $\mathbf{Y}$ is polynomially bounded.

In terms of security, we require a notion of 1-ciphertext simulation security in the sense that the simulator is required to be able to "program" the output of computation on the encrypted input of a challenge ciphertext. More specifically, there exists a simulator that can simulate a secret key nsk_L and a ciphertext $\mathsf{nct}^\star$ for input $\mathbf{v}^\star$ given only L and $L(\mathbf{v}^\star) + \mathbf{Y}$, where $\mathbf{Y}$ is sampled from η. However, in the secret key setting, adversaries cannot produce ciphertexts on their own and we must directly model security when multiple ciphertexts are available. On the other hand, is well know that simulation security is impossible when the number of ciphertexts is unbounded and ciphertexts are sublinearly compact. Instead, we do not require the simulator to "program" the outputs for all encrypted input, it only needs to do so for one challenge ciphertexts, and is given with the actual encrypted inputs for all other ciphertexts — hence the name 1-ciphertext simulation security. Note that this notion is not new, as many works achieve indistinguishability based security via showing such 1-ciphertext simulation security. More precisely, we require

$$\left\{ \ \mathsf{nsk}_L, \ \mathsf{nct}^\star, \ \{\mathsf{nct}_i\}_{i\in[t]} \right\} \approx \left\{ \mathbf{e} \leftarrow \eta \ : \ \mathsf{Sim}\Big(L, \ L\big(\mathbf{x}^\star\big) + \mathbf{e}, \ \{\mathbf{x}_i\}_{i\in[t]}\Big) \right\} \ .$$

where nsk_f and $\mathsf{nct}^\star$ are the challenge key and ciphertext and every nct_i is an honestly generated ciphertext for an arbitrary input x_i, which is given to the simulator.

Compared to noisy linear function encryption by Agrawal [Agr18a], our notion differs in three points: First, we parametrize the notion by the noise distribution η, while Agrawal's notion is parametrized by a bound on the decryption error and distributions restricting the adversary's challenge messages. Secondly, we only require weak correctness. And thirdly, we consider simulation-security, whereas Agrawal defines indistinguishability-based security.

Construction from PHFE and noise generator. There is a simple construction of an η-noisy secret-key linear FE scheme if there is a PHFE scheme for a function class $\mathcal{G}$ and a noise generator G in the same class whose outputs are indistinguishable to η. Take for example our PHFE scheme from bilinear map (Theorem 4) for computing multilinear cubic polynomials $g(\mathbf{z}_1, \mathbf{z}_2, \mathbf{z}_3)$ in $\mathbb{Z}_p$ with $\mathbf{z}_1$ public and $\mathbf{z}_2, \mathbf{z}_3$ private. Assume there is a family of noise generators and seed distributions $(G, \mathcal{D}^{sd}) \leftarrow \mathcal{NG}$ observing the same structure, whose output distribution $G(\mathbf{s}_1, \mathbf{s}_2, \mathbf{s}_3)$ (with $(\mathbf{s}_1, \mathbf{s}_2, \mathbf{s}_3 \leftarrow \mathcal{D}^{sd})$ is indistinguishable to η when $\mathbf{s}_1$ is made public. We can construct η-noisy linear FE as follows:

– To encrypt a vector $\mathbf{v}$, the encryptor samples a seed $(\mathbf{s}_1, \mathbf{s}_2, \mathbf{s}_3)$ and encrypts $\mathbf{z}_1 = \mathbf{s}_1$ as the public input, and $\mathbf{z}_2 = (\mathbf{v}||\mathbf{s}_2), \mathbf{z}_3 = \mathbf{s}_3$ as the private inputs.
– To generate a key for a function f, it generates a key for the function $g(\mathbf{z}_1, \mathbf{z}_2, \mathbf{z}_3) = L(\mathbf{v}) + \mathbf{Y}$ where $\mathbf{Y} = G(\mathbf{s}_1, \mathbf{s}_2, \mathbf{s}_3)$.

Decryption clearly recovers $L(\mathbf{v}) + \mathbf{Y}$, where by the property of the noise generator G, $\mathbf{Y}$ is distributed indistinguishably to η. For the 1-ciphertext simulation security

to hold, we correspondingly need the underlying PHFE to satisfy 1-ciphertext simulation security (defined similarly that a simulator can "program" the output for a single challenge ciphertext), which our construction achieves. Finally, observe that the ciphertexts are sublinear compact, as long as G has superlinear stretch. We provide a formal description and proofs of the construction in the full version [LM18].

Back to Constant-Degree FE Recall that we want to use a noisy linear FE scheme to perform the linear half decryption on the output ciphertext CT_f, $L_{\mathrm{dec}}(\mathsf{CT}_f, \mathbf{s})$, and obtain $\mathbf{y} + 2\mathbf{e} + 2\mathbf{Y}$ (think of η as a distribution over $2\mathbf{Y}$). We still face two challenges:

- **privacy:** Our PHFE from bilinear maps (and all known sublinearly compact degree-d FE from degree-d multilinear maps) only allows decryption if outputs reside in a polynomially sized range. (This is because computation is performed in the exponent, and outputs are extracted via brute force discrete logarithm.) This means $\mathbf{y} + 2\mathbf{e} + 2\mathbf{Y}$ must be polynomially bounded. However, as argued in the introduction, a polynomially-bounded $\mathbf{Y}$ cannot hide $\mathbf{e}$ entirely. But revealing $\mathbf{e}$ at even *one* coordinate potentially reveals information about $\mathbf{x}$.
- **integrity:** How can we ensure that only output ciphertexts CT_f for legitimate constant-degree polynomials f can be decrypted? To ensure that, we would like to give out a noisy linear FE secret key nsk for the function $L_{\mathrm{dec}}(\mathsf{CT}_f, \star)$ and ciphertext nct encrypting the HE secret key $\mathbf{s}$. However, the key generator has no idea what CT_f is.

For the privacy problem, we weaken the requirements on the noise generators, formulating PFG, so that outputs are polynomially bounded and $\mathbf{e}$ is guaranteed to be partially hidden; then, we manage the leakage on $\mathbf{e}$ to still achieve meaningful security. For the integrity problem, we follow the approach of [AR17, Agr18a] of using special (1-time) HE that has a special decryption equation. We elaborate in the next section.

4.2 Weak and Leaky Constant-Degree FE

Let's first consider the privacy problem: How to manage leakage of the value $\mathbf{e}_i$'s at a few coordinates i's? Since $\mathbf{e}_i$ does depend on $\mathbf{x}$, some information of $\mathbf{x}$ is for sure leaked. Hence, we aim for what is the best possible: ensuring that revealing $\mathbf{e}$ at *a few* coordinates translates to revealing $\mathbf{x}$ at *a few* coordinates, if the function computed has *small locality*. We show that this can be done, and construct constant-degree FE with (1-key) *weak and leaky* 1-ciphertext simulation security. Roughly speaking, it guarantees that for every distribution of $f \leftarrow \mathcal{FN}$ and every distribution of $\mathbf{x} \leftarrow \mathcal{X}$, the secret key sk_f for f and the ciphertext $\mathsf{CT}_\mathbf{x}$ for $\mathbf{x}$ can be simulated a simulator Sim using the output $y = f(x)$, as well as the value of $\mathbf{x}$ at a few coordinates. In addition, in the multi-ciphertext setting, the adversaries also see a set of additional ciphertexts $\mathsf{CT}_{\mathbf{x}_i}$ for arbitrary inputs $\mathbf{x}_i$,

and the simulator is required to simulate them given the actual inputs $\mathbf{x}_i$. More precisely, there is correlated random variables K and x^* representing the set of leaked coordinates and their values, such that $|x^*| = |K| = o(\lambda)$ and

$$\{ x, \ sk_f, \mathsf{CT}_x, \ \{\mathsf{CT}_{x_i}\} \ \} \ \approx \ \{ \ x, \ \mathsf{Sim} \, (\ (x^*, K), \ f, \ y = f(x), \ \{x_i\} \ ) \} \ ,$$
$$\text{where } (x^*, K) \leftarrow \mathsf{Fix}, \text{ and } x \leftarrow \mathcal{X}|_{x^*, K} \ .$$

In other words, given sk_f, CT_x, and many other ciphertexts the encrypted input x appears random up to a few coordinates being fixed and the output being y.

We now give some intuition on why weak and leaky simulation security is achievable. Assume that $\mathbf{Y} + \mathbf{e}$ reveals a few coordinates of $\mathbf{e}$, say with index set J, and hides all other coordinates. We carefully analyze what information $\mathbf{e}_J$ depends on: if the function computed has small locality, output elements in J depend only on *a few* input elements at coordinates J'. Suppose an ideal case where the HE scheme satisfies the following properties:

HE properties:
1. *Preserving locality:* the homomorphic evaluation preserves this locality and $\mathbf{e}_J$ depends only on ciphertexts $\mathbf{c}_{J'}$ encrypting $\mathbf{x}_{J'}$,
2. *Preserving entropy:* revealing information related to a few ciphertexts $\mathbf{c}_{J'}$ only reduces the entropy of $\mathbf{s}$ by a small amount, and
3. *Robustness:* the HE scheme used is *robust to small leakage of the secret key*.

We can assert that ciphertexts encrypting other coordinates of $\mathbf{x}$ outside J' remain hiding, and hence only a few coordinates of $\mathbf{x}$ at J' are leaked.

For the above argument to go through, we need a slightly stronger version of the flawed-smudging property: For any B-bounded noise vector distribution $\chi = e(\mathcal{R})$, where the noise $\mathbf{e}$ is the output of a local function over another distributional secret $\mathbf{w} \leftarrow \mathcal{R}$, there is a correlated random variable I such that

$$\{ \ I, \ \mathbf{w}, \ \mathbf{Y} + e(\mathbf{w}) \ \} \approx \{ \ I, \ \mathbf{w}', \ \mathbf{Y} + e(\mathbf{w}) \ \}, \text{ where } \mathbf{w}' \leftarrow \chi|_{\mathbf{w}_I, I} \ .$$

This means given $\mathbf{Y} + e(\mathbf{w})$, only a few coordinates of $\mathbf{w}$ get fixed and leaked. In our construction, $\mathbf{w}$ depends on the input $\mathbf{x}$, the HE secret $\mathbf{s}$, and the noises used originally for encrypting $\mathbf{x}$. The above property then allows us to bound what information of them is leaked through $\mathbf{e}$. We further show that this stronger flawed-smudging property is in fact implied by the normal flawed-smudging property that is agnostic of how $\mathbf{e}$ is generated.

Let us now consider the integrity problem: How can we ensure only CT_f for the right f is decrypted? The works of [AR17, Agr18b] presented HE schemes whose ciphertexts $\mathbf{c}_\mathbf{x}$ consists of $\mathbf{A}, \mathsf{hCT}_\mathbf{x}$, where $\mathbf{A}$ is public and independenent of the input $\mathbf{x}$ (e.g., $\mathbf{A}$ could be LWE matrices, or RLWE scalars) and only $\mathsf{hCT}_\mathbf{x}$ depends on $\mathbf{x}$. Furthermore, homomorphic evaluation operates on $\mathbf{A}$ and $(\mathbf{A}, \mathsf{hCT}_\mathbf{x})$ respectively to obtain $\mathbf{A}_f$ and hCT_f, and decryption does:

4. *Special decryption equation:*

$$\mathbf{s}_f = L_{\text{dec}}(\mathbf{A}_f, \mathbf{s}), \qquad \mathsf{hCT}_f + \mathbf{s}_f = f(\mathbf{x}) + 2\mathbf{e}^{11} \qquad (\text{mod } p)$$

We can view $\mathbf{s}_f$ as a decryption key for f and it is computed from $\mathbf{s}$ independently of hCT_f! We can now ensure integrity as follows:

– Fix $\mathbf{A}$ at set-up time. This means the same $\mathbf{A}$ is reused for all HE ciphertexts.
– The key generator publishes a noisy linear FE key nsk for $L_{\text{dec}}(\mathbf{A}_f, \star)$
– The encryptor publishes $\mathsf{hCT}_{\mathbf{x}}$ encrypting $\mathbf{x}$ using secret $\mathbf{s}$ and generates a noisy linear FE ciphertext nct encrypting $\mathbf{s}$.
– The decryptor decrypts $\mathsf{nct}, \mathsf{nsk}$ to obtain $\mathbf{s}_f + 2\mathbf{Y}$, and computes hCT_f from hCT, from which $\mathbf{y} + 2\mathbf{e} + 2\mathbf{Y}$ is revealed.

Note that since $\mathbf{A}$ is fixed and reused for all HE ciphertext, each secret key $\mathbf{s}$ can only be used once. This is not a problem as the encryptor can sample a fresh secret key $\mathbf{s}$ for each encryption.

Instantiating the HE scheme. The question now is whether there is a HE scheme that simultaneously has the special decryption equation (property 4) and is robust to leakage (properties 1-3). The schemes in [AR17, Agr18a] unfortunately are complicated and we do not know how to analyze their robustness to leakage. Nevertheless, we manage to construct a HE scheme satisfying all 4 properties, based on the simple HE scheme by [BV11] from LWE. We sketch our design. First, it was shown in [GKPV10, AKPW13], that the LWE assumption is robust, in the sense that when the LWE secret $\mathbf{s}$ comes from a small domain (e.g., $[-1, 0, 1]^\lambda$), the hardness of LWE holds as long as $\mathbf{s}$ has sufficient entropy. Thus, it is easy to observe that the HE schemes of [BV11, BGV12] are robust. Furthermore, the simple BV-scheme without relinearization, which can already handle constant-degree computations, also satisfies properties *1)* and *2)*.

However, the simple-BV scheme does not have the special decryption equation. Inspired by [AR17, Agr18a], we use a recursive construction to homomorphically evaluate the BV-decryption itself similar to bootstrapping, but for a different purpose. In slightly more details, we can decompose the BV evaluation-and-decryption procedure $\mathsf{HE.Dec}(\mathbf{s}, \mathsf{HE.Eval}(f, \mathsf{hCT}))$ into a public part Pub that does not depend on the secret key $\mathbf{s}$ and a private part Priv that depends on $\mathbf{s}$.

$$\mathsf{CT}_f = \mathsf{Pub}(f, \mathsf{hCT}, \mathbf{A}) \qquad \mathbf{s}_f = \mathsf{Priv}(f, \mathbf{A}, \mathsf{hCT}, \mathbf{s})$$
$$\mathsf{CT}_f + \mathbf{s}_f = f(\mathbf{x}) + 2\mathbf{e}$$

A wishful thinking is giving out noisy linear FE key nsk for $\mathsf{Priv}(f, \mathbf{A}, \star, \star)$ and ciphertext nct for $(\mathsf{hCT}, \mathbf{s})$, to enable computing $\mathbf{s}_f$. This does not work as Priv has degree d in $\mathbf{s}$ and degree $d-1$ in hCT, where d is the degree of f. The high

---

[11] The schemes in [AR17, Agr18a] has more complicated decryption equation, where the decryption noise is of form $\sum_i p_i \mathbf{e}_i$ where $\{p_i\}$ is a set of increasing moduli. Here we omit this complexity.

degree in $\mathbf{s}$ can be dealt with as the encryptor can compute all degree d monomials in $\mathbf{s}$ and encrypt them, and there are only n^d of them where $n = |\mathbf{s}| = \text{poly}(\lambda)$. But, the same cannot be applied to hCT which is long (length $S^{1-\epsilon}$, where S is the output length of L) and encrypting even the quadratic monomials would make the ciphertexts non-compact. However, the good news is that the degree in hCT is $d - 1$ — one less than the degree of the computation f. Therefore, by recursively encrypting $((\mathsf{hCT}, 1) \otimes (\mathbf{s}, 1) \otimes (\mathbf{s}, 1))$ in a ciphertext hCT' using an independent secret key $\mathbf{s}'$, we can compute $\mathbf{s}_f$ by homomorphically evaluating Priv on hCT' in degree $d - 1$ and then decrypt. The key observation is that the new private computation $\mathsf{Priv}'(\mathsf{Priv}, \mathbf{A}', \mathsf{hCT}', \mathbf{s}')$ now has only degree $d - 2$ in hCT'. Thus, we can recursively reduce the degree of private computation, till we obtain a scheme whose Priv is linear in its ciphertext hCT and degree 2 in its secret key $\mathbf{s}$. Hence,

$$\mathsf{Priv}(f, \mathbf{A}, \mathsf{hCT}, \mathbf{s}) = L_{f,\mathbf{A}}((\mathsf{hCT}, 1) \otimes (\mathbf{s}, 1) \otimes (\mathbf{s}, 1))$$

where the total the number of monomials to be encrypted is $|\mathsf{hCT}|n^2$, keeping sublinear compactness. In summary, our weak and leaky FE for local constant degree computation operates as follows:

- Fix $\mathbf{A}$ at set-up time.
- The key generator publishes a noisy linear FE key nsk for $L_{f,\mathbf{A}}$.
- Tne encryptor publishes a ciphertext hCT encrypting $\mathbf{x}$ under secret key $\mathbf{s}$ using our recursively constructed HE scheme, and generates a noisy linear FE ciphertext nct encrypting $(\mathsf{hCT}, 1) \otimes (\mathbf{s}, 1) \otimes (\mathbf{s}, 1)$.
- The decryptor decrypts $\mathsf{nct}, \mathsf{nsk}$ to obtain $\mathbf{s}_f + 2\mathbf{Y}$ and computes $\mathsf{hCT}_f = \mathsf{Pub}(f, \mathsf{hCT}, \mathbf{A})$, from which $\mathbf{y} + 2\mathbf{e} + 2\mathbf{Y}$ is revealed.

The above description is simplified; please see the full paper [LM18] for a formal description and analysis of our constant degree FE scheme.

4.3   New Bootstrapping to FE for NC_1

We next present a new bootstrapping technique to FE for NC^1 from weak and leaky constant-degree FE. Our bootstrapping follows the same paradigm as previous works [Lin16b, LV16, Lin17, AS17, LV17]: it uses a randomized encoding [IK02, AIK04] to transform an NC^1 computation $g(v)$ into a simple constant-degree constant-locality polynomial $\hat{g}(v; r)$, and uses a constant locality PRG to supply pseudorandom coins $r = \mathsf{PRG}(\mathsf{Seed})$ needed for the randomized encoding. The fact that the underlying constant-degree FE is weak and leaky means both the input v, as well as the PRG seed Seed may be fixed and leaked at a few coordinates. To deal with this, we introduce a new primitive called *Bit-Fixing Homomorphic Sharing* in order to make the original computation g robust.

Our (T, t_1, t_2)-bit-fixing homomorphic sharing resembles the recent new concept of Homomorphic Secret Sharing (HSS) [BGI15] in syntax, but differs in security and efficiency requirements. It enables compiling a single computation

$g(v)$ into a collection of computations $o_1 = h_1(x_1), \ldots, o_T = h_T(x_T)$ that operate on a secret sharing $x_1, \ldots, x_T$ of the original input v, and from the collection of output shares $o_1, \ldots, o_T$, the original output $g(v)$ can be reconstructed. Security ensures that the original input v remains hidden, given all output shares $o_1, \ldots, o_T$ and a subset of t_2 input shares. Moreover, the security is robust to a few t_1 bits in the input shares being fixed. In terms of efficiency, we allow the output share size to scale with the size of the computation g, however, it should not depend on the number of computations to be preformed — in other words, the shares are reusable.

In comparison, HSS shares need to be succinct and output reconstruction needs to be simple, which are not required here. In terms of security, HSS is secure against an adversary seeing a subset of the input shares only. From these input shares, the adversaries can always derive the corresponding output shares, but not all output shares. In contrast, our bit-fixing homomorphic sharing is secure against adversaries seeing all output shares. Note, however, HSS with additive reconstruction i.e., $o = \sum_i o_i$, does satisfy this stronger security, since the adversaries knowing the output o can easily reverse sample the missing additive output shares [12].

We give a construction of bit-fixing homomorphic sharing BF from multi-key FHE with threshold decryption as constructed in [MW16], which roughly works as follows:

– BFsetup samples a CRS crs for the multi-key FHE.
– BFshare shares a string v as follow: It additively shares v into $v = \text{ss}_1 \oplus \ldots \oplus \text{ss}_T$, generates T key-pairs (PK_i, sk_i) of the multi-key FHE scheme, and encrypts the ith share ss_i under PK_i to obtain ciphertext CT_i. It additionally samples a PRF key K_i. Finally, the i'th share is set to

$$x_i = \left\{ \{\text{CT}_i, \text{PK}_i\}_{i \in [T]}, sk_i, K_i \right\}$$

– BFeval on input crs, x_i, i, g evaluates g on the i'th share as follows: It homomorphically evaluates the function g on all ciphertexts $\text{CT}_1, \ldots, \text{CT}_T$ obtaining CT_f. By properties of the multi-key FHE scheme, this output ciphertexts can be decrypted in a distributed way using each secret key sk_i independently. Hence, the i'th output share o_i is set to the value decrypted from CT_g by sk_i. (The decryption procedure of MKFHE is actually randomized. BFeval uses the PRF K_i to generate the pseudorandom coins).
– BFdec reconstructs the final output o from $o_1, \cdots, o_T$ using the reconstruction procedure of the multi-key FHE.

Security of this scheme follows simply from the security of the multi-key FHE scheme and the fact that less than T additive shares ss_i reveal nothing about v.

Next, to construct FE for NC_1, instead of using our weak and leaky constant-degree FE to compute the randomized encodings $\{\mathsf{RE}(g_j, v \; ; \; \mathsf{PRG}_j(\mathsf{Seed}))\}_j$ for each output bit $g_j(v)$ directly, where $\mathsf{PRG}_i(\mathsf{Seed})$ denotes the j'th chunck of output

---

[12] Thanks Yuval Ishai and Elette Boyal for pointing this out.

bits of PRG, we compute the randomized encodings $\{\mathsf{RE}(\mathsf{BFeval}, (\mathrm{crs}, x_i, i, g_j);$ $\mathsf{PRG}_{i,j}(\mathsf{Seed}))\}_{i\in[T],j}$ for evaluating each g_j on each input share x_i. By the weak and leaky security of constant-degree FE, only a few coordinates of its encrypted input, here $\{(\mathrm{crs}x_i, i, g)\}$ and Seed, are leaked. Small leakage on $\{(\mathrm{crs}x_i, i, g)\}$ alone is harmless, as the security of bit-fixing homomorphic sharing ensures that the original input v would remain hidden under such leakage.

However, small leakage on Seed is problematic. Consider a typical local PRG where every output bit depends on $O(1)$ randomly chosen seed bits. Since PRG maps $S^{1-\alpha}$ bits to S bits where S is proportional the size of g, each seed bit Seed_k influences a large number, S^ϵ on average, of output bits. If Seed_k is leaked, all these output bits are no longer pseudorandom — call them corrupted. In turn, all the randomized encodings that use these output bits are no longer hiding, which may leak all input shares x_i. To circumvent this, instead of having only a single set of shares $\{x_i\}_{i\in[T]}$, we will have $M = S^{1-\alpha}$ sets of shares $\{x_i^t\}_{t\in[M],i\in[T]}$. We divide the output bits of g into M chunks, each containing S^α bits, and the t'th chunk is computed using the t'th set of input shares as described above. Why does this help? Suppose that the locations of the corrupted PRG output bits are distributed randomly. Since there are only about $\mathrm{poly}(\lambda)S^\alpha$ corrupted output bits, whereas way more $M = S^{1-\alpha}$ chunks, with overwhelming probability, no chunk ends up using more than λ corrupted PRG output bits. As a result, for each set of input shares $\{x_i^t\}$, at most λ input shares are leaked, and the security of bit fixing homomorphic sharing kicks in again, and hence v is hidden. To ensure that corrupted PRG output bits indeed distribute randomly, we apply a random permutation π to the output of the PRG. In other words, the i'th pseudorandom bit is the $\pi(i)$'th PRG output bit.

In summary, our FE scheme for NC^1 depth Dep proceeds as follows: DFE is our weak and leaky FE for local constant degree computation.

$\mathsf{FE.Setup}(1^\lambda)$: Generate a DFE master secret key DMSK, and a CRS for the bit-fixing homomorphic sharing scheme crs. Output $\mathsf{MSK} = (\mathsf{DMSK}, \mathrm{crs})$.

$\mathsf{FE.KeyGen}(\mathsf{MSK}, g)$: g is a NC_1 function with input-length N, output-length S, and depth Dep. Assume w.l.o.g. that every output bit g_i is computable in some fixed polynomial size $= \mathrm{poly}(\lambda)$.[13]

- Generate a polynomial f as follows:
  - Divide the output bits of g into $M = S^{1-\alpha}$ (assume for convenience that M divides S) consecutive chunks $I_1, \ldots, I_M$, where chunk I_j includes output bits $(j-1)S/M + 1, \ldots, jS/M$. For every $j \in [M]$, let $g_{I_j} = \{g_k\}_{k\in I_j}$ denote the collection of circuits that computes output bits in chunk I_j.
  - For every $j \in [M]$ and $i \in [\lambda]$, let D_i^j be the circuit that on input the i'th share x_i^j of the j'th sharing $\mathbf{x}^j$ of v, homomorphically evaluates g_{I_j}, i.e.,

$$D_i^j(x_i^j) = \mathsf{BFeval}(\mathrm{crs}, x_i^j, i, g_{I_j}) = o_i^j .$$

---

[13] If not, one can always use garbled circuits to turn g into another circuit where every output bit is computable in size $\mathrm{poly}(\lambda)$, at the cost of increasing the size, input length, and output length of the circuit by a multiplicative $\mathrm{poly}(\lambda)$ factor.

- Choose a random permutation $\pi\colon [\lambda] \times [M] \times [\phi] \to [\lambda M \phi]$. For every $j \in [M]$ and $i \in [\lambda]$, let f_i^j be the following function:

$$f_i^j(x_i^j, \mathsf{Seed}) = \mathsf{REnc}\big(D_i^j, x_i^j \;;\; \mathsf{PRG}_{\Pi_i^j}(\mathsf{Seed})\big).$$

Above, $\mathsf{PRG}_{\Pi_i^j}(\mathsf{Seed})$ contains PRG output bits at locations $\{\pi(i,j,k)\}_{k\in[\phi]}$ determined by the random permutation π and is sufficiently long for supplying the random coins needed for computing the randomized encoding.
Finally, set

$$f\left(\left\{\mathbf{x}^j = \{x_i^j\}_{i\in[\lambda]}\right\}_{j\in[M]}, \mathsf{Seed}\right) := \left\{f_i^j(x_i^j, \mathsf{Seed})\right\}_{j,i}.$$

- Generate a DFE secret key of f, $\mathrm{D}sk \leftarrow \mathsf{DFE.KeyGen}(\mathsf{DMSK}, f)$.
  This can be done since by the efficiency of the bit-fixing homomorphic sharing and randomized encoding, the input length and size of f is $N' = |\{x_i^j\}| + |\mathsf{Seed}| = \mathrm{poly}(\lambda, s)\lambda M + \mathrm{poly}(\lambda)S^{1-\alpha} = \mathrm{poly}(\lambda)S^{1-\alpha}$ and $S' = |f_i^j|\lambda M = \mathrm{poly}(\lambda)S$. Since the AIK randomized encoding algorithm REnc and PRG both have constant locality, f also has constant locality ℓ. Moreover, over the field $\mathbb{Z}_2$, it has at most degree ℓ.

Output $sk = \mathrm{D}sk$.

$\mathsf{FE.Enc}(\mathsf{MSK}, v)$: On input $\mathsf{MSK} = (\mathsf{DMSK}, \mathrm{crs})$ and $v \in \{0,1\}^N$, do:

- For every $j \in [M]$, generate the j'th BF sharing of v, $\mathbf{x}^j = \{x_i^j\}_{i\in[\lambda]} \leftarrow \mathsf{BFshare}(\mathrm{crs}, v)$.
- Sample randomly a PRG seed Seed.
- Encrypt $X = (\{\mathbf{x}^j\}_j, \mathsf{Seed})$ using DFE, $\mathsf{DCT} \leftarrow \mathsf{DFE.Enc}(\mathsf{DMSK}, X)$.

Output $\mathsf{CT} = \mathsf{DCT}$.

$\mathsf{FE.Dec}(sk, \mathsf{CT})$: On input $sk = \mathrm{D}sk$ and $\mathsf{CT} = \mathsf{DCT}$, do

- Decrypt the DFE ciphertext DCT with the secret key $\mathrm{D}sk$ to obtain $y = f(X) = \mathsf{DFE.Dec}(\mathrm{D}sk, \mathsf{DCT})$.
- Parse $y = \{y_i^j\}$, and for every $j \in [M]$ and $i \in [\lambda]$, decode y_i^j using REval to obtain $o_i^j = \mathsf{REval}(y_i^j)$.
- For every $j \in [M]$, decode the output shares $\{o_i^j\}_{i\in[\lambda]}$ to obtain the actual output $u^j = \mathsf{BFdec}(\mathrm{crs}, \{o_i^j\})$.

Output $u = \{u^j\}$.

Correctness of the construction can be shown as follows: By the correctness of DFE, we have

$$y = f(X) = \left\{y_i^j = f_i^j(x_i^j, \mathsf{Seed})\right\}_{j,i},$$
$$y_i^j = \mathsf{REnc}(D_i^j, x_i^j \;;\; \mathsf{PRG}_{\Pi_i^j}(\mathsf{Seed})) .$$

By the correctness of RE, we have that

$$o_i^j = \mathsf{REval}(y_i^j) = D_i^j(x_i^j) = \mathsf{BFeval}(\mathrm{crs}, x_i^j, i, g_{I_j}) = o_i^j \ .$$

By the correctness of BF, we have that $u^j = g_{I_j}(v)$.

In the full version [LM18], we formally prove that the above construction is a sublinearly compact secret key FE scheme for NC_1 satisfying standard indistinguishability-based security, which implies $i\mathcal{O}$.

References

AB15.   Benny Applebaum and Zvika Brakerski. Obfuscating circuits via composite-order graded encoding. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *Theory of Cryptography*, pages 528–556, Berlin, Heidelberg, 2015. Springer.

AFV11. Shweta Agrawal, David Mandell Freeman, and Vinod Vaikuntanathan. Functional encryption for inner product predicates from learning with errors. In Dong Hoon Lee and Xiaoyun Wang, editors, *Advances in Cryptology – ASIACRYPT 2011*, pages 21–40, Berlin, Heidelberg, 2011. Springer.

AG11. Sanjeev Arora and Rong Ge. New algorithms for learning in presence of errors. In *Automata, Languages and Programming - 38th International Colloquium, ICALP 2011, Zurich, Switzerland, July 4-8, 2011, Proceedings, Part I*, pages 403–415, 2011.

AGIS14a. Prabhanjan Ananth, Divya Gupta, Yuval Ishai, and Amit Sahai. Optimizing obfuscation: Avoiding Barrington's theorem. In *ACM CCS*, pages 646–658, 2014.

AGIS14b. Prabhanjan Ananth, Divya Gupta, Yuval Ishai, and Amit Sahai. Optimizing obfuscation: Avoiding Barrington's theorem. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, CCS '14, pages 646–658, New York, NY, USA, 2014. ACM.

Agr17a. Shweta Agrawal. Stronger security for reusable garbled circuits, general definitions and attacks. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology – CRYPTO 2017*, pages 3–35, Cham, 2017. Springer International Publishing.

Agr17b. Shweta Agrawal. Stronger security for reusable garbled circuits, general definitions and attacks. In *CRYPTO*, pages 3–35, 2017.

Agr18a. Shweta Agrawal. New methods for indistinguishability obfuscation: Bootstrapping and instantiation. Cryptology ePrint Archive, Report 2018/633, 2018. https://eprint.iacr.org/2018/633.

Agr18b. Shweta Agrawal. Personal communication and a previous version of eprint report 2018/633, 2018.

AIK04. Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Cryptography in NC0. In *45th Annual IEEE Symposium on Foundations of Computer Science*, pages 166–175, 10 2004.

AJS18. Prabhanjan Ananth, Aayush Jain, and Amit 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.

AKPW13. Joël Alwen, Stephan Krenn, Krzysztof Pietrzak, and Daniel Wichs. Learning with rounding, revisited. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013*, pages 57–74, Berlin, Heidelberg, 2013. Springer.

AL16. Benny Applebaum and Shachar Lovett. Algebraic attacks against random local functions and their countermeasures. In *Proceedings of the Forty-eighth Annual ACM Symposium on Theory of Computing*, STOC '16, pages 1087–1100, New York, NY, USA, 2016. ACM.

AR17. Shweta Agrawal and Alon Rosen. Functional encryption for bounded collusions, revisited. In Yael Kalai and Leonid Reyzin, editors, *Theory of Cryptography*, pages 173–205, Cham, 2017. Springer International Publishing.

AS17. Prabhanjan Ananth and Amit Sahai. Projective arithmetic functional encryption and indistinguishability obfuscation from degree-5 multilinear maps. In *EUROCRYPT*. Springer, 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.

BBKK18.   Boaz Barak, Zvika Brakerski, Ilan Komargodski, and Pravesh K. Kothari. Limits on low-degree pseudorandom generators (or: Sum-of-squares meets program obfuscation). In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018*, pages 649–679, Cham, 2018. Springer International Publishing.

BCFG17.   Carmen Elisabetta Zaira Baltico, Dario Catalano, Dario Fiore, and Romain Gay. Practical functional encryption for quadratic functions with applications to predicate encryption. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology – CRYPTO 2017*, pages 67–98, Cham, 2017. Springer International Publishing.

BFM14.   Christina Brzuska, Pooya Farshim, and Arno Mittelbach. Indistinguishability obfuscation and uces: The case of computationally unpredictable sources. In *CRYPTO*, pages 188–205, 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 *Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part I*, pages 565–596, 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+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, 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19-23, 2001, Proceedings*, volume 2139 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2001.

BGI15.   Elette Boyle, Niv Gilboa, and Yuval Ishai. Function secret sharing. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology - EUROCRYPT 2015*, pages 337–367, Berlin, Heidelberg, 2015. Springer.

BGK+14a.   Boaz Barak, Sanjam Garg, Yael Tauman Kalai, Omer Paneth, and Amit Sahai. Protecting obfuscation against algebraic attacks. In *CRYPTO*, pages 221–238, 2014.

BGK+14b.   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, *Advances in Cryptology – EUROCRYPT 2014*, pages 221–238, Berlin, Heidelberg, 2014. Springer.

BGMZ18.   James Bartusek, Jiaxin Guan, Fermi Ma, and Mark Zhandry. Return of GGH15: provable security against zeroizing attacks. In *Theory of Cryptography - 16th International Conference, TCC 2018, Panaji, India, November 11-14, 2018, Proceedings, Part II*, pages 544–574, 2018.

BGV12.   Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, ITCS '12, pages 309–325, New York, NY, USA, 2012. ACM.

BHJ+19. Boaz Barak, Samuel B. Hopkins, Aayush Jain, Pravesh Kothari, and Amit Sahai. Sum-of-squares meets program obfuscation, revisited. In *Advances in Cryptology - EUROCRYPT 2019 - 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19-23, 2019, Proceedings, Part I*, pages 226–250, 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.

BPR15. Nir Bitansky, Omer Paneth, and Alon Rosen. On the cryptographic hardness of finding a nash equilibrium. In *FOCS*, 2015.

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

BR14a. Zvika Brakerski and Guy N. Rothblum. Virtual black-box obfuscation for all circuits via generic graded encoding. In *TCC*, pages 1–25, 2014.

BR14b. Zvika Brakerski and Guy N. Rothblum. Virtual black-box obfuscation for all circuits via generic graded encoding. In Yehuda Lindell, editor, *Theory of Cryptography*, pages 1–25, Berlin, Heidelberg, 2014. Springer.

BS02. Dan Boneh and Alice Silverberg. Applications of multilinear forms to cryptography. 324, 11 2002.

BSW11. Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In *Theory of Cryptography*, pages 253–273. Springer, 2011.

BTVW17. Zvika Brakerski, Rotem Tsabary, Vinod Vaikuntanathan, and Hoeteck Wee. Private constrained PRFs (and more) from LWE. In Yael Kalai and Leonid Reyzin, editors, *Theory of Cryptography*, pages 264–302, Cham, 2017. Springer International Publishing.

BV11. Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In *2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*, pages 97–106, 10 2011.

BWZ14. Dan Boneh, David J. Wu, and Joe Zimmerman. Immunizing multilinear maps against zeroizing attacks. *IACR Cryptology ePrint Archive*, 2014:930, 2014.

CCL18. Yi-Hsiu Chen, Kai-Min Chung, and Jyun-Jie Liao. On the complexity of simulating auxiliary input. *IACR Cryptology ePrint Archive*, 2018:171, 2018.

CEMT09. James Cook, Omid Etesami, Rachel Miller, and Luca Trevisan. Goldreich's one-way function candidate and myopic backtracking algorithms. In Omer Reingold, editor, *Theory of Cryptography*, pages 521–538, Berlin, Heidelberg, 2009. Springer.

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.

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/.

CLT13a.   Jean-Sébastien Coron, Tancrède Lepoint, and Mehdi Tibouchi. Practical multilinear maps over the integers. In *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I*, pages 476–493, 2013.

CLT13b.   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*, pages 476–493, Berlin, Heidelberg, 2013. Springer.

CLT15a.   Jean-Sebastien Coron, Tancrede Lepoint, and Mehdi Tibouchi. New multilinear maps over the integers. In *CRYPTO*, 2015.

CLT15b.   Jean-Sébastien Coron, Tancrède Lepoint, and Mehdi Tibouchi. New multilinear maps over the integers. In Rosario Gennaro and Matthew Robshaw, editors, *Advances in Cryptology – CRYPTO 2015*, pages 267–286, Berlin, Heidelberg, 2015. Springer.

CM01.   Mary Cryan and Peter Bro Miltersen. On pseudorandom generators in NC0. In Jiří Sgall, Aleš Pultr, and Petr Kolman, editors, *Mathematical Foundations of Computer Science 2001*, pages 272–284, Berlin, Heidelberg, 2001. Springer.

CVW18.   Yilei Chen, Vinod Vaikuntanathan, and Hoeteck Wee. GGH15 beyond permutation branching programs: Proofs, attacks, and candidates. In *Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part II*, pages 577–607, 2018.

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

DGG+16b.   Nico Döttling, Sanjam Garg, Divya Gupta, Peihan Miao, and Pratyay Mukherjee. Obfuscation from low noise multilinear maps. Cryptology ePrint Archive, Report 2016/599, 2016. https://eprint.iacr.org/2016/599.

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 *EUROCRYPT*, 2014.

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, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings*, volume 7881 of *Lecture Notes in Computer Science*, pages 1–17. Springer, 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 Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA*, pages 40–49. IEEE Computer Society, 2013.

GGH15a.   Craig Gentry, Sergey Gorbunov, and Shai Halevi. Graph-induced multilinear maps from lattices. In *TCC*, pages 498–527, 2015.

GGH15b.   Craig Gentry, Sergey Gorbunov, and Shai Halevi. Graph-induced multi-linear maps from lattices. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *Theory of Cryptography*, pages 498–527, Berlin, Heidelberg, 2015. Springer.

GGHR14.   Sanjam Garg, Craig Gentry, Shai Halevi, and Mariana 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.

GGHZ16.   Sanjam Garg, Craig Gentry, Shai Halevi, and Mark Zhandry. Functional encryption without obfuscation. In *Theory of Cryptography - 13th International Conference, TCC 2016-A, Tel Aviv, Israel, January 10-13, 2016, Proceedings, Part II*, pages 480–511, 2016.

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.

GKPV10.   Shafi Goldwasser, Yael Tauman Kalai, Chris Peikert, and Vinod Vaikuntanathan. Robustness of the learning with errors assumption. In *Innovations in Computer Science - ICS 2010, Tsinghua University, Beijing, China, January 5-7, 2010. Proceedings*, pages 230–240, 2010.

GLSW14.   Craig Gentry, Allison B. Lewko, Amit Sahai, and Brent Waters. Indistinguishability obfuscation from the multilinear subgroup elimination assumption. *IACR Cryptology ePrint Archive*, 2014:309, 2014.

GMM+16a.   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 Smith, editors, *Theory of Cryptography*, pages 241–268, Berlin, Heidelberg, 2016. Springer.

GMM+16b.   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 Smith, editors, *Theory of Cryptography: 14th International Conference, TCC 2016-B, Beijing, China, October 31-November 3, 2016, Proceedings, Part II*, pages 241–268, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg.

GMW15.   Romain Gay, Pierrick Méaux, and Hoeteck Wee. Predicate encryption for multi-dimensional range queries from lattices. In Jonathan Katz, editor, *Public-Key Cryptography – PKC 2015*, pages 752–776, Berlin, Heidelberg, 2015. Springer.

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 *CRYPTO*, 2016.

GR07.   Shafi Goldwasser and Guy N. Rothblum. On best-possible obfuscation. In *TCC*, pages 194–213, 2007.

Gri01.   Dima Grigoriev. Linear lower bound on degrees of positivstellensatz calculus proofs for the parity. *Theor. Comput. Sci.*, 259(1-2):613–622, 2001.

GSW13.   Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *CRYPTO*, pages 75–92, 2013.

GVW12.   Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Functional encryption with bounded collusions via multi-party computation. In *Advances in Cryptology–CRYPTO 2012*, pages 162–179. Springer, 2012.

GVW15.   Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Predicate encryption for circuits from LWE. In Rosario Gennaro and Matthew Robshaw, editors, *Advances in Cryptology – CRYPTO 2015*, pages 503–523, Berlin, Heidelberg, 2015. Springer.

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

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 *ASIACRYPT*, pages 715–744, 2016.

HSW14.   Susan Hohenberger, Amit Sahai, and Brent Waters. Replacing a random oracle: Full domain hash from indistinguishability obfuscation. In *EUROCRYPT*, 2014.

IK02.   Yuval Ishai and Eyal Kushilevitz. Perfect constant-round secure computation via perfect randomizing polynomials. In Peter Widmayer, Stephan Eidenbenz, Francisco Triguero, Rafael Morales, Ricardo Conejo, and Matthew Hennessy, editors, *International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 244–256, Berlin, Heidelberg, 2002. Springer.

Imp95.   Russell Impagliazzo. Hard-core distributions for somewhat hard problems. In *FOCS*, pages 538–545, 1995.

JLMS19.   Aayush Jain, Huijia Lin, Christian Matt, and Amit Sahai. How to leverage hardness of constant-degree expanding polynomials over $\mathbb{R}$ to build $i\mathcal{O}$. In *Advances in Cryptology - EUROCRYPT 2019 - 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19-23, 2019, Proceedings, Part I*, pages 251–281, 2019.

JLS19.   Aayush Jain, Huijia Lin, and Amit Sahai. Removing the need block-local prgs to build io. *IACR Cryptology ePrint Archive*, 2019, 2019.

JP14.   Dimitar Jetchev and Krzysztof Pietrzak. How to fake auxiliary input. In *TCC*, pages 566–590, 2014.

JS18.   Aayush Jain and Amit Sahai. How to leverage hardness of constant-degree polynomials over r to build io. *IACR Cryptology ePrint Archive*, 2018:973, 2018.

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

KS99.   Aviad Kipnis and Adi Shamir. Cryptanalysis of the HFE public key cryptosystem by relinearization. In *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, pages 19–30, 1999.

Lin16a.   Huijia Lin. Indistinguishability obfuscation from constant-degree graded encoding schemes. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 28–57. Springer, 2016.

Lin16b.   Huijia Lin. Indistinguishability obfuscation from constant-degree graded encoding schemes. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology – EUROCRYPT 2016*, pages 28–57, Berlin, Heidelberg, 2016. Springer.

Lin17.   Huijia Lin. Indistinguishability obfuscation from sxdh on 5-linear maps
            and locality-5 prgs. In *CRYPTO*, pages 599–629. Springer, 2017.

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

LSS14.   Adeline Langlois, Damien Stehlé, and Ron Steinfeld. GGHLite: More
            efficient multilinear maps from ideal lattices. In Phong Q. Nguyen and
            Elisabeth Oswald, editors, *Advances in Cryptology – EUROCRYPT 2014*,
            pages 239–256, Berlin, Heidelberg, 2014. Springer.

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*, pages 630–660,
            Cham, 2017. Springer International Publishing.

LV16.   Huijia Lin and Vinod Vaikuntanathan. Indistinguishability obfuscation
            from ddh-like assumptions on constant-degree graded encodings. In *FOCS*,
            pages 11–20. IEEE, 2016.

LV17.   Alex Lombardi and Vinod Vaikuntanathan. Limits on the locality of pseu-
            dorandom generators and applications to indistinguishability obfuscation.
            In Yael Kalai and Leonid Reyzin, editors, *Theory of Cryptography*, pages
            119–137, Cham, 2017. Springer International Publishing.

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

MST03.   Elchanan Mossel, Amir Shpilka, and Luca Trevisan. On e-biased generators
            in NC0. In *44th Annual IEEE Symposium on Foundations of Computer
            Science, 2003. Proceedings.*, pages 136–145, 10 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.

MT10.   Ueli M. Maurer and Stefano Tessaro. A hardcore lemma for computational
            indistinguishability: Security amplification for arbitrarily weak prgs with
            optimal stretch. In *TCC*, pages 237–254, 2010.

MW16.   Pratyay Mukherjee and Daniel Wichs. Two round multiparty computation
            via multi-key FHE. In *EUROCRYPT*, pages 735–763, 2016.

MZ18.   Fermi Ma and Mark Zhandry. The mmap strikes back: Obfuscation and
            new multilinear maps immune to CLT13 zeroizing attacks. In *Theory of
            Cryptography - 16th International Conference, TCC 2018, Panaji, India,
            November 11-14, 2018, Proceedings, Part II*, pages 513–543, 2018.

O'N10.   Adam O'Neill. Definitional issues in functional encryption. *IACR Cryptol-
            ogy ePrint Archive*, 2010:556, 2010.

OW14.   Ryan O'Donnell and David Witmer. Goldreich's PRG: Evidence for
            near-optimal polynomial stretch. In *2014 IEEE 29th Conference on Com-
            putational Complexity (CCC)*, pages 1–12, June 2014.

PST14a.   Rafael Pass, Karn Seth, and Sidharth Telang. Indistinguishability obfus-
            cation 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.

PST14b.   Rafael Pass, Karn Seth, and Sidharth Telang. Indistinguishability obfus-
            cation from semantically-secure multilinear encodings. In Juan A. Garay

and Rosario Gennaro, editors, *Advances in Cryptology – CRYPTO 2014*, pages 500–517, Berlin, Heidelberg, 2014. Springer.

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

Sch08. Grant Schoenebeck. Linear level lasserre lower bounds for certain k-csps. In *49th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2008, October 25-28, 2008, Philadelphia, PA, USA*, pages 593–602, 2008.

SW05. Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In *Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005, Proceedings*, pages 457–473, 2005.

SW14. Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In David B. Shmoys, editor, *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 475–484. ACM, 2014.

Wol02. Christopher Wolf. "hidden field equations" (HFE) - variations and attacks. Master's thesis, Universität Ulm, December 2002. http://www.christopher-wolf.de/dpl.

Zim15. Joe Zimmerman. How to obfuscate programs directly. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology - EUROCRYPT 2015*, pages 439–467, Berlin, Heidelberg, 2015. Springer.