

# Pseudo Flawed-Smudging Generators and Their Application to Indistinguishability Obfuscation

Huijia Lin

Christian Matt

*University of California, Santa Barbara*  
{[rachel.lin](mailto:rachel.lin), [cmatt](mailto:cmatt)}@cs.ucsb.edu

## Abstract

We introduce *Pseudo Flawed-smudging Generators* (PFGs). A PFG is an expanding function whose outputs  $\mathbf{Y}$  satisfy a weak form of pseudo-randomness. Roughly speaking, for some polynomial bound  $B$ , and every distribution  $\chi$  over  $B$ -bounded noise vectors, it guarantees that the distribution of  $(\mathbf{e}, \mathbf{Y} + \mathbf{e})$  is indistinguishable from that of  $(\mathbf{e}', \mathbf{Y} + \mathbf{e})$ , where  $\mathbf{e} \leftarrow \chi$  is a random sample from  $\chi$ , and  $\mathbf{e}'$  is another independent sample from  $\chi$  conditioned on agreeing with  $\mathbf{e}$  at *a few*,  $o(\lambda)$ , coordinates. In other words,  $\mathbf{Y}$  “hides”  $\mathbf{e}$  at *all but a few* coordinates. We show that assuming LWE and the existence of constant-locality Pseudo-Random Generators (PRGs), there is a construction of IO from 1) a PFG that has polynomial stretch and polynomially bounded outputs, and 2) a Functional Encryption (FE) scheme able to compute this PFG. Such FE can be built from degree  $d$  multilinear map if the PFG is computable by a degree  $d$  polynomial.

Toward basing IO on bilinear maps, inspired by [Ananth et. al. Eprint 2018], we further consider PFGs with partial public input — they have the form  $g(\mathbf{x}, \mathbf{y})$  and satisfy the aforementioned pseudo flawed-smudging property even when  $\mathbf{x}$  is public. When using such PFGs, it suffices to replace FE with a weaker notion of *partially hiding* FE (PHFE) whose decryption reveals the public input  $\mathbf{x}$  in addition to the output of the computation. We construct PHFE for polynomials  $g$  that are *quadratic* in the private input  $\mathbf{y}$ , but have up to *polynomial* degree in the public input  $\mathbf{x}$ , subject to certain size constraints, from the SXDH assumption over *bilinear* map groups.

Regarding candidates of PFGs with partial public input, we note that the family of cubic polynomials proposed by Ananth et. al. can serve as candidate PFGs, and can be evaluated by our PHFE from bilinear maps. Toward having more candidates, we present a transformation for converting the private input  $\mathbf{x}$  of a constant-degree PFG  $g(\mathbf{x}, \mathbf{y})$  into a public input, by hiding  $\mathbf{x}$  as noises in LWE samples, provided that  $\mathbf{x}$  is sampled from a LWE noise distribution and  $g$  satisfies a stronger security property.

# Contents

<b>1</b>	<b>Introduction</b>	<b>4</b>
1.1	Our Results in More Detail	6
1.2	Our Techniques	9
1.3	Related Works	12
1.4	Outline of the Paper	14
<b>2</b>	<b>Preliminaries</b>	<b>14</b>
2.1	Notation and Basic Definitions	14
2.2	Learning with Errors	15
2.3	Asymmetric Bilinear Maps and the SXDH Assumption	16
2.4	Pseudorandom Generators and Pseudorandom Functions	16
2.5	Symmetric Encryption	17
2.6	Indistinguishability Obfuscation	18
2.7	Randomized Encodings	18
2.8	Functional Encryption	19
2.8.1	Secret-Key Functional Encryption	19
2.8.2	FE for P/poly, $NC^1$ , and Inner Products, and Compactness	20
2.9	(Fully) Homomorphic Encryption	21
2.9.1	Threshold Multi-Key FHE	23
<b>3</b>	<b>Pseudo Flawed-Smudging Generators</b>	<b>24</b>
3.1	Definitions	24
3.2	Properties of (Flawed-)Smudging Distributions	27
3.2.1	Bounds on the Smudging Property	27
3.2.2	Preservation Under Addition of Independent Values	28
3.2.3	Mixtures of (Flawed-)Smudging Distributions	29
3.2.4	Composition of Flawed-Smudging Distributions	30
3.2.5	Smudging and Independence Implies Flawed-Smudging	32
3.2.6	Hiding Function Inputs	35
<b>4</b>	<b>Partially-Hiding Functional Encryption</b>	<b>38</b>
4.1	PHFE for Polynomials of Degree 3	39
4.2	PHFE for Polynomials with Polynomial Degree	45
<b>5</b>	<b>Noisy Secret-Key Linear Functional Encryption</b>	<b>50</b>
5.1	Definitions	50
5.2	Construction from PHFE and Noise Generator	52
5.3	Instantiating the PHFE and Noise Generator	55
5.4	Construction from FE and Noise Generator	57
<b>6</b>	<b>Functional Encryption for Constant Degree Polynomials</b>	<b>58</b>
6.1	Review of the Simple-BV HE Scheme	58
6.2	HE Schemes with Linear Private Evaluation-Decryption Functions	62
6.3	Construction of DFE	66
6.3.1	Property of DFE w.r.t. a Special Purpose Function Distribution	74

<b>7</b>	<b>Functional Encryption for <math>\text{NC}^1</math> and Transformation to IO</b>	<b>76</b>
7.1	Bit-Fixing Homomorphic Sharing . . . . .	76
7.1.1	Construction from Threshold Multi-Key FHE . . . . .	77
7.2	Construction of FE for $\text{NC}^1$ . . . . .	79
7.3	From Secret-Key Ciphertext-Compact FE to IO . . . . .	87
<b>8</b>	<b>Weakening Requirements on Flawed-Smudging Distributions</b>	<b>87</b>
8.1	Flawed-Smudging Distributions with Small Probability . . . . .	88
8.2	Using Weaker Distributions in Previous Constructions . . . . .	88
8.3	Amplifying Security . . . . .	90
	<b>Acknowledgments</b>	<b>93</b>

# 1 Introduction

Indistinguishability obfuscation (IO), first defined in the seminal work of Barak et. al. [BGI<sup>+</sup>01], aims to obfuscate functionally equivalent programs into indistinguishable ones while preserving functionality. IO is an extraordinarily powerful object that has been shown to enable a large set of new cryptographic applications.

The first-generation IO constructions [GGH<sup>+</sup>13, BR14, BGK<sup>+</sup>14, PST14, AGIS14, GLSW15, Zim15, AB15, GMM<sup>+</sup>16, DGG<sup>+</sup>16] rely on polynomial-degree *multilinear maps* or *graded encodings*. An  $L$ -linear map [BS03] 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 — so far, vulnerabilities [CHL<sup>+</sup>15, CGH<sup>+</sup>15, MSZ16, CGH17, ADGM17] were demonstrated against all known candidates [CLT13, LSS14, GGH15, CLT15]. Of course, this does not mean that the resulting IO constructions are insecure; in particular, the construction of [GMM<sup>+</sup>16] is formally shown to withstand all existing attacks.

A line of recent works [Lin16, LV16, Lin17, AS17] aimed at finding the minimal degree of multilinear maps sufficient for constructing IO, and has successfully reduced the required degree to  $L = 3$ . A key ingredient in these second-generation constructions are PRGs with *small locality*<sup>1</sup>. They showed that to construct IO, it suffices to have multilinear maps with degree matching exactly the locality of the PRG [Lin16, AS17], or even a 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. This raises the following natural question:

*Are there different types of simple PRGs with weak security that are useful for building IO from trilinear maps? or even bilinear maps?*

**Flawed-Smudging Generators.** Toward answering these questions, we propose *Pseudo Flawed-smudging Generators* (PFGs) that have much weaker security requirements than pseudo-randomness and different structural properties from local PRGs. More specifically, they are polynomially expanding functions from  $n$  input elements to  $m = n^{1+\alpha}$  output elements (where  $n, m$  are parameterized by the security parameter  $\lambda$ ), satisfying a weak form of pseudo randomness that we call *pseudo flawed-smudging* (described shortly below). To construct IO from  $d$ -linear maps, we need PFGs that are computable by low degree  $d$  polynomials over  $\mathbb{Z}_p$  (with no restriction on locality) and have polynomially bounded outputs (i.e., every output element is an integer of polynomial magnitude). As such, they can be computed in the exponent of  $d$ -linear map groups, and their outputs can be extracted via brute force discrete logarithm.

The *pseudo flawed-smudging* property guarantees that the outputs of a PFG (on inputs from a specific distribution) can “smudge”, or flood, a small noise vector, at *all but a few*,  $o(\lambda)$ , coordinates. More precisely, its output distribution is indistinguishable to a, so-called, *flawed-smudging distribution*  $\mathbf{Y} \leftarrow \mathcal{Y}$ , satisfying that for some polynomial  $B$ , and every  $B$ -bounded noise vector distribution  $\mathbf{e} \leftarrow \chi$ ,  $\mathbf{Y} + \mathbf{e}$  “hides” the value of the noise vector  $\mathbf{e}$  at *all but a few* coordinates: There is a random variable  $I$  correlated with  $\mathbf{e}, \mathbf{Y}$ , representing a small,  $|I| = o(\lambda)$ ,

---

<sup>1</sup>A function has locality  $\ell$  if every output element depends on at most  $\ell$  input elements

subset of “compromised” coordinates, so that the joint distribution of  $(I, \mathbf{e}, \mathbf{Y} + \mathbf{e})$  is statistically close to that of  $(I, \mathbf{e}', \mathbf{Y} + \mathbf{e})$ , where  $\mathbf{e}'$  is a fresh new sample from  $\chi$  conditioned on agreeing with  $\mathbf{e}$  at coordinates in  $I$  (i.e.,  $e'_i = e_i$  for all  $i \in I$ ):

$$\{ I, \mathbf{e}, \mathbf{Y} + \mathbf{e} \} \approx_s \{ I, \mathbf{e}', \mathbf{Y} + \mathbf{e} \}, \text{ where } \mathbf{e}' \leftarrow \chi|_{\mathbf{e}, I}.$$

In short, given  $\mathbf{Y} + \mathbf{e}$ , the noise vector  $\mathbf{e}$  remains randomly distributed, up to a few coordinates being fixed.

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, the smudging distributions used in the literature usually have *super-polynomially* large output elements (for instance, a discrete Gaussian with super-polynomial standard deviation, or a uniform distribution over a consecutive super-polynomial sized support). A sample  $\mathbf{Y}$  from such distributions can hide a small noise vector  $\mathbf{e}$  *entirely* at all coordinates (with overwhelming probability), in the sense that  $(\mathbf{e}, \mathbf{Y} + \mathbf{e}) \approx (\mathbf{e}', \mathbf{Y} + \mathbf{e})$  for a completely independent  $\mathbf{e}' \leftarrow \chi$ . In fact, to hide the noise vectors  $\mathbf{e}$  entirely, it is *necessary* that  $\mathbf{Y}$  is super-polynomially large. This highlights the key rationale behind the definition of flawed-smudging distributions — when the smudging distribution is polynomially bounded, it inevitably “reveals” the noise vector  $\mathbf{e}$  at some coordinates with non-negligible probability.

The fact that PFG has polynomially small outputs is crucial for evaluating it in the exponent of multilinear map groups and extracting the outputs. Leveraging this, if degree 3 (or generally  $d$ ) PFGs exist, we give a new approach for building IO from trilinear maps (or  $d$ -linear map). Natural candidate (degree  $d$ ) PFGs are random (degree  $d$ ) multivariate polynomials with *small* inputs and coefficients. Evaluating such polynomials over  $\mathbb{Z}_p$  with large modulus does not trigger wrap-around and the outputs are guaranteed to be small.

**Toward Bilinear Maps.** Ideally, we would like to have degree 2 PFGs, to obtain IO from bilinear maps. However, it has already been shown that random Multivariate Quadratic (MQ) polynomials with small inputs and coefficients are not secure [BHK<sup>+</sup>18]. The work of [AJKS18] proposed a variant of the natural candidate that is a degree 3 multilinear polynomial  $g(\mathbf{x}, \mathbf{y}, \mathbf{z})$  over three input vectors, where the first input  $\mathbf{x}$  can be made public without hurting the weak pseudo-randomness properties of  $g$ 's output. Thanks to the fact that computation on the private inputs  $\mathbf{y}, \mathbf{z}$  is only quadratic, they construct a Functional Encryption (FE) scheme for evaluating such functions, in the generic bilinear map model.

Inspired by their proposal and the notion of partial hiding predicate encryption of [GVW15], we consider the generalization — *partially-hiding FE*, which is implicit in [GVW12]; they can evaluate functions  $g(\mathbf{x}, \mathbf{y})$  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  $\mathbf{c}$  outputs 1, then it corresponds to attribute-based encryption.

Using just bilinear map groups where the SXDH assumption holds, we implement partially-hiding FE supporting the computation of  $g = q(f(\mathbf{x}), \mathbf{y})$  that first performs a complex, up to polynomial degree, computation  $f$  on the public input, followed by a simple, quadratic, computation  $q$  with the private input, and  $f$  is a formula with width bounded by  $\max(|\mathbf{x}|, |\mathbf{y}|)^2$ . We can use this partially-hiding FE to evaluate PFGs computable by such function  $g$ , including

the candidate proposed in [AJKS18]. To extend the repertoire of potential candidates, we describe a transformation for turning  $g(\mathbf{x}, \mathbf{y})$  with no public input into  $h(\mathbf{c}, \mathbf{y}')$  with a public input by hiding  $\mathbf{x}$  in LWE samples  $\mathbf{c}$ . This method is implicit in the candidate of [AJKS18].

## 1.1 Our Results in More Detail

We describe our results in two parts; Part 1 contains results appeared in our previous Eprint version and Part 2 is new in this version<sup>2</sup>.

**Part 1: FE from degree- $d$  PFG and  $d$ -linear map.** Leveraging the simple structure of PFGs, we construct secret-key functional encryption schemes for computing polynomial-sized  $\text{NC}^1$  circuits with sublinearly compact ciphertexts whose sizes grow polynomially in the security parameter  $\lambda$  and input length  $N$ , and sublinearly in the size  $S$  of the circuits computed.<sup>3</sup> Our schemes satisfy standard 1-key (fully-selective) indistinguishability security, based on the assumptions summarized in the following theorem.

**Theorem 1.1** (Informal). *Assume the LWE assumption, the SXDH assumption over asymmetric  $d$ -linear map groups of order  $p$ , the existence of a family of constant-locality PRGs (with mild structural properties described below), and a family of Pseudo Flawed-smudging Generators computable by degree  $d$  polynomials over  $\mathbb{Z}_p$  with polynomially bounded outputs. There is a construction of secret-key functional encryption schemes for computing polynomial-sized circuits in  $\text{NC}^1$ , with sublinearly compact ciphertexts and 1-key fully selective indistinguishability security.*

*If all assumptions and primitives are subexponentially secure, so are the functional encryption schemes.*

The above theorem relies on a family of PRGs mapping  $n$  bits to  $n^{1+\alpha}$  bits for an arbitrarily small constant  $\alpha$ , where every PRG is defined by a predicate  $P$  and an input-output dependency graph  $G$ , such that the  $i$ 'th output bit  $y_i = P(sd_{G(i)})$  is computed by evaluating the predicate  $P$  on a subset of seed bits  $sd_{G(i)}$  specified by  $G(i)$ . We require 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].

Previous works [AJ15, BV15, LPST16b, LPST16a, BNPW16, KNT18] showed that functional encryption schemes with sublinearly compact ciphertexts and subexponential security imply indistinguishability obfuscation. We thus get the following corollary.

**Corollary 1.2** (Informal). *Assume the same assumptions as in Theorem 1.1, all with subexponential security. Then, there is a construction of subexponentially secure indistinguishability obfuscation for polynomial-sized circuits.*

<sup>2</sup>The results in Part 1 are concurrent and independent with the work by Ananth et. al. [AJKS18], and the results in Part 2 are follow-up to their work.

<sup>3</sup>In [BV15, AJ15], the notion of compactness requires the encryption time to be  $\text{poly}(\lambda, N)S^{1-\epsilon}$ , where  $N$  and  $S$  are respectively the input length and size of the computation. However, in later works [LPST16b, LPST16a, BNPW16, BLP17], a more relaxed notion of compactness was considered which only requires the ciphertext size to be mildly compact, and allows encryption time to depend polynomially on  $S$ . We use the relaxed notion in this work.

Our techniques for achieving [Theorem 1.1](#) gives a new way of constructing compact functional encryption, that instead of performing the computation  $\mathbf{y} = f(\mathbf{x})$  in the exponent of multi-linear map groups, performs the computation using homomorphic encryption, and uses multi-linear maps only to decrypt; for most HE schemes, decryption involves a *linear* operation, which already reveals the output perturbed by a small noise  $\mathbf{y} + 2\mathbf{e}$ . In order to ensure that only the output is revealed, the multi-linear map is used to additionally evaluate PFGs and add the generated smudging noises  $\mathbf{Y}$  to the decryption output  $\mathbf{y} + 2\mathbf{e} + 2\mathbf{Y}$  to hide  $\mathbf{e}$ . Therefore, the degree of the PFG determines the degree of the multi-linear map. More specifically, we use FE for computing degree- $d$  polynomials, or degree- $d$  FE, for short to evaluate PFG, which in turn can be based on degree- $d$  multilinear map [[AS17](#), [Lin17](#)].

*Preliminary Study of PFGs.* A natural class of candidate degree- $d$  PFGs is random degree- $d$  multivariate polynomials  $g(\mathbf{x})$  with small inputs and coefficients. More specifically, for every  $l$ , the  $l$ 'th output noise is computed by  $g^l(\mathbf{x}_{G(l)})$ , where the function  $g^l$ , the input  $\mathbf{x}$ , and the input-output dependency graph  $G$  can all be sampled from some distribution. For instance, we can consider constant degree  $d = O(1)$  polynomials  $g^l(\mathbf{x}^1, \mathbf{x}^2, \dots, \mathbf{x}^d) = \sum_{i_1, \dots, i_d} c_{i_1, \dots, i_d} x_{i_1}^1 \dots x_{i_d}^d$  with inputs and coefficients sampled from some polynomially bounded distribution. The advantage of having small inputs and coefficients is that, when the degree is a constant, the outputs are kept small. The disadvantage is that the computation never triggers wrap-around modulo  $p$ , which may be beneficial for security. Indeed, the degree 2 polynomials  $g^l(\mathbf{x}, \mathbf{y}) = \sum_{i,j} c_{i,j} x_i y_j$  with small inputs and coefficients from distributions proposed in recent works, including an earlier version of this work, were broken [[BHK<sup>+</sup>18](#)].

We further study properties of flawed-smudging distributions. First, we show that distributions in the following class are flawed-smudging:  $\mathcal{Y}$  is the product  $\mathcal{Y} = \mathcal{Y}_1 \times \dots \times \mathcal{Y}_m$  of independent distributions  $\mathcal{Y}_i$  for sampling individual smudging noises such that the statistical distance  $\delta(\mathcal{Y}_i, \mathcal{Y}_i + e)$  between  $\mathcal{Y}_i$  and  $\mathcal{Y}_i$  shifted by a small noise  $e \in [-B, B] \cap \mathbb{Z}$  is bounded by a sufficiently small inverse polynomial. (For example,  $\mathcal{Y}_i$  could be a polynomially wide Gaussian distribution over  $\mathbb{Z}$ , or a uniform distribution over a polynomially sized consecutive integer interval.) Secondly, we show that the flawed-smudging property is preserved under addition with an independent distribution, and under convex combinations. This property can help us combine multiple PFG candidates, or a PFG candidate and an independent function to enhance security.

**Part 2: Toward Bilinear Maps — Partially-Hiding Functional Encryption and Weakening PFGs.** Toward the goal of relying only on bilinear maps, it is important to extend the class of PFGs that can be evaluated using bilinear maps as much as possible. As mentioned above, inspired by [[GVW15](#), [AJKS18](#)], we consider Partially-Hiding FE (PHFE) that computes functions  $g(\mathbf{x}, \mathbf{y})$  in a way revealing the output and also the public input  $\mathbf{x}$ , while hiding the private input  $\mathbf{y}$ . (See [Section 4](#) for the formal definition.) We first show that using only bilinear maps, we can have PHFE for polynomials that are quadratic in its private input and linear in its public input.

**Theorem 1.3** (Informal). *Assume the SXDH assumption over asymmetric bilinear map of order  $p$ . There is a construction of secret-key partially-hiding functional encryption schemes for computing polynomials  $g(\mathbf{x}, \mathbf{y}, \mathbf{z})$  over  $\mathbb{Z}_p$  that are multilinear in  $\mathbf{x}$ ,  $\mathbf{y}$ , and  $\mathbf{z}$ , and have polynomially bounded outputs. The encryption time is  $\text{poly}(\lambda)N$ , where  $N$  is the length of the inputs  $N \geq \max(|\mathbf{x}|, |\mathbf{y}|, |\mathbf{z}|)$ .*

Our construction improves that of [[AJKS18](#)] for the same class of polynomials in terms of

assumptions — their scheme is proven secure in the generic bilinear map model, whereas we rely on the SXDH assumption.

Next, building upon the above scheme for degree 3 multilinear polynomial, we construct PHFE for functions  $g$  with up to polynomial degree in the public input  $\mathbf{x}$ , using still bilinear maps. However, we cannot handle the general case, and need to put certain constraints on  $g$ , as described below.

**Theorem 1.4** (Informal). *Assume SXDH over asymmetric bilinear maps of order  $p$ . There is a construction of secret-key partially-hiding functional encryption schemes for computing arithmetic formulas with fan-in 2 multiplication and unbounded fan-in addition over  $\mathbb{Z}_p$  of form:*

- $g(\mathbf{x}, \mathbf{y}, \mathbf{z}) = q(f(\mathbf{x}), \mathbf{y}, \mathbf{z})$ , where  $q$  is multilinear, and  $f$  has logarithmic  $O(\log \lambda)$  multiplicative depth, and the output length of  $g$  and the width of  $f$  are bounded by  $N^2$ , where  $N$  is the input length  $N \geq \max(|\mathbf{x}|, |\mathbf{y}|, |\mathbf{z}|)$ .

The encryption time  $\text{poly}(\lambda)N$ .

Weakening PFGs — 1) Allow Public Input. We can use the above PHFE scheme from bilinear map to evaluate any PFG  $g(\mathbf{x}, \mathbf{y}, \mathbf{z}) = q(f(\mathbf{x}), \mathbf{y}, \mathbf{z})$  that has a public input  $\mathbf{x}$  and satisfies the special form specified in Theorem 1.4; in particular, for a PFG with  $N^{1+\varepsilon}$ -stretch, to satisfy the constraint on the width of  $f$ , it suffices to require that every output noise  $g_l$  is computed by a formula of size  $N^{1-\varepsilon}$  (which bounds the width of  $f$ ).

Weakening PFGs — 2) Transformation for Converting Private Input to Public Input. We now describe a transformation that turns a private input into a public input. Our transformation is inspired by the  $\Delta$ RG candidate of [AJKS18]. As a warm-up, consider a multilinear polynomial  $g(\mathbf{x}, \mathbf{y}, \mathbf{z})$  where all three inputs vectors are private and  $\mathbf{x}$  is sampled from a LWE noise distribution. The key idea is that we can hide  $\mathbf{x}$  in LWE samples  $\mathbf{c}_i = (\mathbf{a}_i, \mathbf{a}_i \mathbf{s}' + x_i) \bmod p$  as the noise terms. Then computing  $g$  translates into computing another function  $h$  where  $x_i$  is replaced with  $\langle \mathbf{c}_i, \mathbf{s} \rangle \bmod p$  for  $\mathbf{s} = (-\mathbf{s}' || 1)$ ,

$$h(\mathbf{c}, \mathbf{y}' = (\mathbf{y} \otimes \mathbf{s}), \mathbf{z}) := \sum_j s_j g(\mathbf{c}_{*,j}, \mathbf{y}, \mathbf{z}) = g(\{\langle \mathbf{c}_i, \mathbf{s} \rangle\}_i, \mathbf{y}, \mathbf{z}) = g(\mathbf{x}, \mathbf{y}, \mathbf{z}) \pmod{p},$$

where  $\mathbf{c}_{*,j}$  is the vector containing the  $j$ 'th element of all LWE samples. By providing the tensor  $\mathbf{y} \otimes \mathbf{s}$  as input, the polynomial  $h$  remains multilinear. Also, note that its computation triggers wrap-around modulo  $p$  due to LWE “decryption”. For  $h$  to be secure at the presence its public input, it means that the output of  $g$  is indistinguishable to a pseudo-smudging distribution, say  $\mathcal{X}$ , even when its first input is hidden in some LWE samples,

$$\{g(\mathbf{x}, \mathbf{y}, \mathbf{z}), \{\mathbf{c}_i = (\mathbf{a}_i, \mathbf{a}_i \mathbf{s} + x_i)\}_i\} \approx \{\Delta \leftarrow \mathcal{X}, \{\mathbf{c}_i = (\mathbf{a}_i, \mathbf{a}_i \mathbf{s} + x_i)\}\}$$

Naturally, this heuristic can be applied to any  $g(\mathbf{x}^1, \dots, \mathbf{x}^d, \mathbf{y}, \mathbf{z})$  that is multilinear in  $d+2$  inputs for some constant  $d = O(1)$ , by defining  $h$  as below with the first  $d$  inputs public

$$\begin{aligned} & h\left(\mathbf{c} = (\mathbf{c}^1, \dots, \mathbf{c}^d), \mathbf{y}' = (\mathbf{y} \otimes \mathbf{s}^{(d/2)}), \mathbf{z}' = (\mathbf{z} \otimes \mathbf{s}^{(d/2)})\right) \\ & := \sum_{j_1, \dots, j_d} s_{j_1} \cdots s_{j_d} g(\mathbf{c}_{*,j_1}^1, \dots, \mathbf{c}_{*,j_d}^d, \mathbf{y}, \mathbf{z}) = g(\mathbf{x}^1, \dots, \mathbf{x}^d, \mathbf{y}, \mathbf{z}), \end{aligned}$$

where every  $\mathbf{x}^k$  is embedded in LWE samples  $\{\mathbf{c}_i^k\}$  with secret  $\mathbf{s}$ , as noises, and  $\mathbf{s}^{(\alpha)}$  denotes the vector obtained after tensoring  $\mathbf{s}$  for  $\alpha$  times. Again,  $h$  is multilinear in  $\mathbf{c}^1, \dots, \mathbf{c}^d, \mathbf{y}', \mathbf{z}'$  and it is secure at the presence of  $\mathbf{c}$ , as long as  $g$  satisfies similarly as above

$$\left\{ g(\mathbf{x}^1, \dots, \mathbf{x}^d, \mathbf{y}, \mathbf{z}), \{\mathbf{c}_i^k = (\mathbf{a}_i^k, \mathbf{a}_i^k \mathbf{s}' + x_i^k)\}_i \right\} \approx \left\{ \Delta \leftarrow \mathcal{X}, \{\mathbf{c}_i^k = (\mathbf{a}_i^k, \mathbf{a}_i^k \mathbf{s}' + x_i^k)\}_i \right\}.$$

Finally, if  $g$  is a constant-degree polynomial satisfying the constraints in Theorem 1.3 or 1.4, then its transformed function  $h$  can be evaluated using the PHFE provided by these theorems.

*Weakening PFGs — 3) Flawed Smudging with only  $1/\text{poly}(\lambda)$  Probability.* Inspired by the notion of  $\Delta\text{RG}$  in [AJKS18], we further weaken our notion of PFG to have outputs indistinguishable from a distribution that is flawed-smudging only with polynomial probability. We show that our techniques for handling PFGs readily extend to handle also these weaker PFGs, which are hence also sufficient for constructing FE and IO.

## 1.2 Our Techniques

**Part 1: FE from degree- $d$  PFG and  $d$ -linear maps.** Toward constructing functional encryption schemes for  $\text{NC}^1$ , we follow the same two-step approach as previous works [Lin16, LV16, Lin17, AS17]: First construct functional encryption schemes for computing constant-degree polynomials, or constant-degree FE for short; then bootstrap constant-degree FE to FE for computing  $\text{NC}^1$  circuits. The key technical difference lies in how the computation of FE is performed. Current constructions of compact (or collusion resistant) FE all perform the computation to be done in the exponent of multilinear map groups. In this work, we explore a different, extremely natural, approach of performing the computation using a Homomorphic Encryption (HE).

*FE via Homomorphic Encryption and Noisy Linear FE.* Several previous works [GVW12, GVW15, GKP<sup>+</sup>13, AR17, Agr18b] have already explored this natural approach in the context of FE. 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 compute a function  $f$  on  $\mathbf{x}$ , the decryptor can homomorphically evaluate  $f$  on  $\mathbf{x}$  and obtain a ciphertext  $\text{ct}_f$  encrypting the output  $\mathbf{y}$ . The challenges are 1) *privacy* — how to decrypt  $\text{ct}_f$  in a secure way that reveals only  $\mathbf{y}$  and hides all other information of  $\mathbf{x}$ , and 2) *integrity* — how to enforce that only ciphertexts associated with a “legitimate” function  $f$  (ones for which secret keys are generated) can be decrypted. The challenges are made harder by the fact that full-fledged HE decryption is a  $\text{NC}^1$  computation. Previous works [GVW12, GKP<sup>+</sup>13, GVW15, AR17, Agr18b] proposed novel ways for achieving them, using a variety of tools from garbled circuits, partial hiding predicate encryption, to noisy linear FE. Unfortunately, the resulting FE schemes fall short in either compactness, or full security, or relying on low degree MMap. Built upon them, our constant-degree FE scheme extends their methods to achieve all three desiderata.

Observe that the decryption of most HE schemes, such as [BV11, BGV12], involves *i)* a simple linear operation, such as  $\langle \text{ct}_f, \mathbf{s} \rangle$  that produces an *approximate* output,  $\mathbf{y} + 2\mathbf{e}$ , perturbed by a small noise vector  $\mathbf{e}$ , *ii)* followed by a threshold function (complex, in  $\text{NC}^1$ ) to remove the noise. It is tempting to ignore the threshold function, and just half-decrypt  $\text{ct}_f$  using a linear FE (encrypting  $\mathbf{s}$ ) to obtain the approximate output. But the noise  $\mathbf{e}$  is *sensitive*, revealing information about the input  $\mathbf{x}$ , the HE secret  $\mathbf{s}$ , and the noises used for generating the original ciphertext encrypting  $\mathbf{x}$ . On the other hand, removing the noise  $\mathbf{e}$  has high complexity. The

works of [AR17, Agr18b] suggest 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  $\mathbf{e}$ . Toward this, Agrawal [Agr18b] introduced the notion of *noisy linear FE*, which adds a *fresh* noise to the decrypted output of every pair of ciphertext and secret key.

However, to implement noisy linear FE, where do these smudging noises come from? If there are  $n$  ciphertexts and  $n$  secret keys, we need  $n^2$  fresh noises. Agrawal [Agr18b] suggests to generate them using a low-degree *noise generator* (similar to our PFG) and a matching low-degree FE to perform both the linear half-decryption and noise generation. This already seems to give a way to implement noisy linear FE using low-degree MMaps. However, a more careful examination reveals a dilemma: Current MMap-based FE schemes can only evaluate polynomials whose outputs are polynomially bounded; but, a polynomially-bounded smudging noise  $\mathbf{Y}$  cannot hide  $\mathbf{e}$  entirely. Agrawal circumvents the problem by implementing low-degree FE supporting super-polynomially sized outputs from a new lattice assumption.

*Weak and Leaky Constant-Degree FE.* We instead ask whether we can, sticking to MMap-based FE and using noises  $\mathbf{Y}$  that only partially hide  $\mathbf{e}$ , still achieve meaningful security. More precisely, we use our Pseudo-Flawed-smudging Generator, which has only polynomially-bounded outputs, and ensures that  $\mathbf{e} + \mathbf{Y}$  hides  $\mathbf{e}$  at *all but a few* coordinates. Since revealing  $\mathbf{e}$  at even one coordinate violates the standard security requirement of FE, we aim for what is the best possible — ensuring that revealing  $\mathbf{e}$  at a few coordinates translates to revealing the input  $\mathbf{x}$  at a few coordinates. By using an HE scheme that is *robust to leakage*, we show that this is possible, and construct constant-degree FE with (1-key) *weak and leaky* simulation security. Roughly speaking, it guarantees that a tuple  $(\text{mpk}, \text{sk}_f, \text{ct}_x)$  consisting of an honestly generated master public key  $\text{mpk}$ , a secret key  $\text{sk}_f$  for a distributional function  $f \leftarrow \mathcal{FN}$ , and a ciphertext  $\text{ct}_x$  for a distributional input  $x \leftarrow \mathcal{X}$ , can be simulated by a simulator  $\text{Sim}$  using the output  $y = f(x)$  of a randomly sampled  $x$  conditioned on its value being fixed at a few coordinates. More precisely, there is a distribution  $\text{Fix}$  over the fixed coordinates  $K$  and values  $x^*$ , such that  $|K| = o(\lambda)$ , and

$$\{ x, \text{mpk}, \text{sk}_f, \text{ct}_x \} \approx \{ x, \text{Sim}((x^*, K), f, y = f(x)) \} ,$$

where  $(x^*, K) \leftarrow \text{Fix}$ , and  $x \leftarrow \mathcal{X}|_{x^*, K}$ .

In other words, given  $\text{mpk}, \text{sk}_f, \text{ct}_x$ , the encrypted input  $x$  appears random up to a few coordinates being fixed, and the output being  $y$ .

HE schemes robust to leakage can be instantiated using the [BV11, BGV12] schemes based on LWE, thanks to the robustness of LWE itself. When the LWE secret  $\mathbf{s}$  comes from a small domain (e.g.,  $\mathbf{s}$  is binary), the hardness of LWE holds as long as  $\mathbf{s}$  has sufficient entropy, and does not necessarily need to be uniformly random [GKPV10, AKPW13]. Furthermore, for the construction of weak and leaky constant-degree FE to go through, we need a slightly stronger version of the flawed-smudging property: Consider a  $B$ -bounded noise vector distribution  $\chi = e(\mathcal{R})$  where the noise  $\mathbf{e}$  is a function over another distributional secret  $\mathbf{w} \leftarrow \mathcal{R}$ ; there is again a correlated random variable  $I$  such that

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

This means given  $\mathbf{Y} + e(\mathbf{w})$ , only a few coordinates of the secret  $\mathbf{w}$  get fixed and leaked. In our construction of constant-degree FE, we use this guarantee to bound what information of the HE input  $\mathbf{x}$  and secret  $\mathbf{s}$  is fixed and leaked through leakage of the noise  $\mathbf{e}$  in the ciphertext obtained via homomorphic evaluation. We further show that the above stronger flawed-smudging property in fact follows from the normal flawed-smudging property that is agnostic of how  $\mathbf{e}$  is generated.

*Bootstrapping from Weak and Leaky Constant-Degree FE.* We next present a new bootstrapping technique to FE for  $\text{NC}^1$  from weak and leaky constant-degree FE. Our bootstrapping follows the same paradigm as previous works [Lin16, LV16, Lin17, AS17, LV17] — it uses a randomized encoding [IK02, AIK04] to transform a  $\text{NC}^1$  computation  $g(v)$  into a simple constant-degree polynomial  $\hat{g}(v; r)$ , and uses a constant locality PRG to supply pseudorandom coins  $r = \text{PRG}(sd)$  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  $sd$  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 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  $o_1 = h_1(x_1), \dots, o_T = h_T(x_T)$  that operates on a secret sharing  $x_1, \dots, x_T$  of the original input  $v$ , and from the collection of outputs  $o_1, \dots, o_T$ , the original output  $g(v)$  can be reconstructed. Security ensures that the original input  $v$  remains hidden, given all output shares  $o_1, \dots, o_T$  and a small subset of input shares (whereas HSS only guarantees that the input remains hidden given a subset of input shares, without the output shares). Moreover, the security is robust to a few bits in the input shares being fixed. We give a construction of bit-fixing homomorphic sharing from multi-key FHE with threshold decryption as constructed in [MW16].

Next, we use the weak and leaky constant-degree FE to compute the randomized encoding of the compiled computations  $\{\hat{h}_i(x_i; r_i)\}$ . Through careful analysis, we show that the weak security of constant-degree FE only leads to a small subset of the computation  $o_i = h_i(x_i)$  being “corrupted”, meaning the input share  $x_i$  is revealed or some bits of  $x_i$  are fixed. It then follows from the security of bit-fixing homomorphic sharing that the original input  $v$  remains hidden.

**Part 2: Partially-Hiding Functional Encryption.** There are known constructions of FE for quadratic polynomials from bilinear maps [Lin17, BCFG17]. It turns out that a simple modification of the scheme by [Lin17] allows for adding a public input and performing a linear computation on it. This yields our PHFE scheme for degree 3 multilinear polynomials  $g(\mathbf{x}, \mathbf{y}, \mathbf{z})$  with  $\mathbf{x}$  public in [Theorem 1.3](#).

Next, toward [Theorem 1.4](#), we build PHFE for computing arithmetic formulas  $g(\mathbf{x}, \mathbf{y}, \mathbf{z}) = q(f(\mathbf{x}), \mathbf{y}, \mathbf{z})$  with complex computation  $f$  on the public input  $\mathbf{x}$ . We design a recursive construction that reduce the depth  $\text{Dep}$  of  $f$  at every level, until reaching the base case where  $g$  is just degree 3 and multilinear. To see how recursion happens, consider an output element  $g_i$  of  $g$ . Since  $q$  is linear in  $f(\mathbf{x})$ , we can write  $g_i$  as  $g_i = \sum_{j \in [k]} \alpha_{i_j} f_{i_j} + \beta_i$ , where coefficients  $\alpha_{i_j}, \beta_i$  depend only on  $\mathbf{y}, \mathbf{z}$ , and  $f_{i_j}$  depends only on  $\mathbf{x}$ . We further decompose the computation of an output element  $f_l$  of  $f$ . Since  $f_l$  has multiplicative depth  $\text{Dep}$ , it can always be computed in degree 2 from intermediate results that have multiplicative depth  $\text{Dep} - 1$ ; written in the form of inner product, we have  $f_l = \langle f_{l,0}^{\text{Dep}-1}, f_{l,1}^{\text{Dep}-1} \rangle$  where the two vectors denote the intermediate results with multiplicative depth  $\text{Dep} - 1$  involved for computing  $f_l$ . Plugging this in, we have that

$$g_i = \sum_{j \in [k]} \alpha_{i_j} \langle f_{i_j,0}^{\text{Dep}-1}, f_{i_j,1}^{\text{Dep}-1} \rangle + \beta_i = \langle \mathbf{U}, \mathbf{V} \rangle,$$

where  $\mathbf{U} = \dots \parallel \alpha_{i_j} f_{i_j,0}^{\text{Dep}-1} \parallel \dots \parallel \beta_i, \quad \mathbf{V} = \dots \parallel f_{i_j,1}^{\text{Dep}-1} \parallel \dots \parallel 1.$

Now we have reduced the computation of  $g$  with depth  $\text{Dep}$  to the computation of vectors  $\mathbf{V}, \mathbf{U}$  with depth  $\text{Dep} - 1$ . This suggests that we can recursively reduce the depth of the computation  $f^{\text{Dep}-1}$  on the public input until it becomes linear, which can then be handled by PHFE for degree 3 multilinear polynomials.

However, one problem is that we cannot leak information of  $\mathbf{U}$ , which in turn reveals information of the private inputs  $\mathbf{y}, \mathbf{z}$  through the values of  $\alpha$  and  $\beta$ . This problem can be solved by instead computing  $\mathbf{U}$  one-time-padded with a random vector  $\mathbf{t}$ , together with the inner product of  $\mathbf{t}$  with  $\mathbf{V}$ . From  $\mathbf{U} + \mathbf{t}$  and  $\langle \mathbf{t}, \mathbf{V} \rangle$ , one can only compute  $g_i$ . The next problem is: where does  $\mathbf{t}$  come from? It can only be provided in the ciphertext, but if so, the ciphertext would be as large as the width of  $f$ . We alleviate this problem by setting  $\mathbf{t} = \mathbf{t}_0 \otimes \mathbf{t}_1$  as the tensor product of much shorter random vectors. Since in our construction  $\mathbf{U} + \mathbf{t}$  and  $\langle \mathbf{t}, \mathbf{V} \rangle$  are computed in the exponent of bilinear map groups, by the SXDH assumption,  $\mathbf{t} = \mathbf{t}_0 \otimes \mathbf{t}_1$  is pseudo-random in the exponent. Now the length of the ciphertext with  $\mathbf{t}_b$  encoded inside only scales with  $\sqrt{\text{width}(f)}$ , which is  $N$  if  $\text{width}(f)$  is bounded by  $N^2$ . This gives the high-level ideas. See [Section 4](#) for the formal construction.

### 1.3 Related Works

As mentioned above, the approach of using a homomorphic encryption scheme to construct functional encryption has already been explored in several works [[GVW12](#), [GKP<sup>+</sup>13](#), [GVW15](#), [AR17](#), [Agr18a](#)]. As discussed before, the challenge are twofold: 1) *privacy* — decrypt a ciphertext  $\text{ct}_f$  encrypting output  $\mathbf{y}$  securely revealing only  $\mathbf{y}$ , and 2) *integrity* — enforce that only ciphertexts for “legitimate” function  $f$  (ones for which secret keys are generated) can be decrypted. Below, we briefly discuss techniques in previous works.

Gorbunov, Vaikuntanathan, and Wee [[GVW12](#)] give a bootstrapping theorem from a FE scheme able to 1) perform homomorphic evaluation of a function  $f$  on public HE ciphertext  $\text{ct}$ , followed by 2) full HE decryption in  $\text{NC}^1$  to FE for  $P$ . The starting point FE is essentially a PHFE scheme, though they did not explicitly define it. The bootstrapping works by letting the FE for  $P/\text{poly}$  encrypt the input  $\mathbf{x}$  using a HE scheme, and then uses the PHFE to perform both the public HE homomorphic evaluation and the private HE decryption, which guarantees both privacy and integrity. However, this PHFE is for an very complex computation.

The work of Goldwasser et al. [[GKP<sup>+</sup>13](#)] took a different approach to perform homomorphic evaluation and decryption, using respectively attribute-based encryption and garbled circuits (with the HE secret  $\mathbf{s}$  hard-coded). The former ensures integrity while the latter ensures privacy. However, since each garbled circuit can only be used to decrypt a single HE ciphertext, to compute a function with  $M$  output elements, the ciphertext must include  $M$  garbled circuits. Thus, their FE scheme has non-compact ciphertexts,<sup>4</sup> which are insufficient for constructing IO.

Gorbunov, Vaikuntanathan, and Wee [[GVW15](#)] made the crucial observation that for all known HE schemes, decryption corresponds to computing an inner product followed by a threshold function. Furthermore, there are lattice-based constructions of predicate encryption schemes for threshold of inner product [[AFV11](#), [GMW15](#)]. Building upon techniques from latter works, they constructed predicate encryption schemes for  $P/\text{poly}$ , which is weaker than functional encryption in the sense that it only guarantees “one-sided” security, that is, privacy of inputs is only guaranteed if the secret key released is for a function that evaluates to the zero vector on

---

<sup>4</sup>In their language, the FE scheme handles bounded collusion, where the ciphertext size scales with the number of secret keys released.

the inputs. Otherwise, given a secret key that evaluates to a non-zero vector, the attacker would not only learn the output of the function, but also some HE decryption noises. In this case, their scheme does not guarantee security; in fact, concrete attacks have been demonstrated [Agr17]. In the context of FE, we would need FE for threshold of inner products, which is only known under IO or poly-degree multilinear maps.

In the work of [AR17], to ensure privacy of HE decryption, they use an FE scheme to perform linear HE 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. Moreover, they also use a new approach to ensure integrity. Instead of relying on primitives, such as, attribute based encryption or PHFE, to ensure integrity as in [GVW12, GKP<sup>+</sup>13, GVW15]. They employ a special HE scheme whose decryption equation has 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  $\mathbf{s}$ .

**Comparison with the work by Agrawal [Agr18a].** Following [AR17], to obtain compact ciphertexts, Agrawal [Agr18b] (*an early version of [Agr18a] was shared with us by the author*) proposed the approach of using a noise generator to generate  $\mathbf{Y}$ . As an abstraction of that, she 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 she proposes a constant degree FE scheme supporting super-polynomially large outputs from a new assumption on NTRU Rings. In this work, we 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. In the recent updated version [Agr18a], Agrawal adds that her construction is compatible with the approach of [AJKS18] using  $\Delta\text{RG}$  with polynomially large outputs and weak security, and later amplify the security of FE in a black-box way. (*This update is concurrent to our work.*)

**Comparison with the work by Ananth et. al. [AJKS18].** Ananth et al. [AJKS18] construct IO from (subexponentially-secure) bilinear maps, LWE, block-locality 3 PRGs, and a new type of randomness generator, called *perturbation resilient generator* ( $\Delta\text{RG}$ ). A  $\Delta\text{RG}$ , introduced in the new work [AJKS18], is a polynomially expanding function, satisfying that for every integer vector  $e \in \mathbb{Z}^\ell$  with coordinates bounded by some polynomial, efficient distinguishers can only distinguish  $\Delta\text{RG}(sd)$  and  $\Delta\text{RG}(sd) + e$  with advantage at most  $1 - 1/\text{poly}(\lambda)$ . Using these tools they construct FE for computing degree-3 polynomials, and then apply a known bootstrapping theorem [LT17] to obtain IO. For their construction of degree-3 FE, they need the  $\Delta\text{RG}$  to be computable by a *cubic multilinear* polynomial  $g(\mathbf{x}, \mathbf{y}, \mathbf{y})$  where  $\mathbf{x}$  is public. They further construct a FE scheme for computing these functions in the generic bilinear map model.

We now compare our notion of PFGs with their  $\Delta\text{RG}$ s. 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. PFGs ensure that given  $\mathbf{Y} + \mathbf{e}$ , only a few coordinates of  $\mathbf{e}$  are compromised, and the rest remain hidden. On the other hand, a  $\Delta\text{RG}$  ensures that all coordinates are simultaneously hidden with some polynomial probability.

Besides the use of different weak notions of randomness generator, other differences include: *i)* We rely on constant-locality PRGs with mild structural properties, while they use block-locality

3 PRGs. *ii)* We rely on the SXDH assumption over bilinear pairing groups, while they show security in the generic bilinear map model.

In terms of techniques, both works start with constructing some weak notions of FE: We construct FE for constant-degree polynomials that may leak a small portion of the input, whereas Ananth et al. construct FE for degree 3 polynomials that bounds the adversarial advantage only by  $1 - 1/\text{poly}(\lambda)$ . They then use the dense model theorem to argue that with probability  $1/\text{poly}(\lambda)$  their FE scheme hides the encrypted input entirely (and with probability  $1 - 1/\text{poly}(\lambda)$ , it may leak the encrypted input). 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, we rely 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”.

*The results discussed in the above comparison are concurrent and independent — they corresponds to Part 1 in our results, described in Section 1.1. After their work and a previous version of this work appeared on Eprint, inspired by their work, we extended our results, described in Part 2 in Section 1.1, in the following aspects: 1) we define PHFE and construct it for special functions with high degree in public input and quadratic in private input from SXDH on bilinear maps, 2) we consider PFGs with constant degree in the public input and quadratic in private input, and describe a transformation for converting private inputs into public ones. 3) we show that weaker PFGs for distributions that are only flawed-smudging with polynomial probability, are already sufficient for our construction.*

## 1.4 Outline of the Paper

We review some notation and standard cryptographic notions that we use in the paper in Section 2. In Section 3, we define pseudo flawed-smudging generators (PFGs) and prove some properties of flawed-smudging distributions. In Section 4, we define partially-hiding FE (PHFE) and provide constructions for some special classes of functions. In Section 5, we give a definition for noisy secret-key linear FE (NFE) and show how to construct it from PHFE and PFGs. Section 6 describes the construction of a functional encryption scheme for constant degree polynomials, which makes use of a NFE scheme. In Section 7, we construct a functional encryption scheme for  $\text{NC}^1$  using our FE scheme from Section 6 and additional tools, including bit-fixing homomorphic sharing, which we introduce in Section 7.1. In Section 8, we finally discuss how to weaken the requirements of PFGs in our construction.

## 2 Preliminaries

### 2.1 Notation and Basic Definitions

We denote by  $\mathbb{Z}$  the set of integers and by  $\mathbb{N}$  the set of nonnegative integers. For  $n, p \in \mathbb{N}$ ,  $[n] := \{1, \dots, n\}$ , and  $[n]_p$  denotes the value  $n$  reduced modulo  $p$ . For a distribution  $\mathcal{D}$ ,  $x \leftarrow \mathcal{D}$  denotes that  $x$  is sampled according to  $\mathcal{D}$ , for a probabilistic algorithm  $A$ ,  $y \leftarrow A(x)$  denotes running  $A$  on input  $x$  and assigning the output to  $y$ , and for a finite set  $S$ ,  $x \leftarrow S$  denotes assigning a uniformly random value from  $S$  to  $x$ . We use the following notation to denote the distribution that samples  $x_i \leftarrow \mathcal{D}_i$  for  $i \in [n]$  and then outputs  $f(x_1, \dots, x_n)$ :

$$\{x_1 \leftarrow \mathcal{D}_1, \dots, x_n \leftarrow \mathcal{D}_n : f(x_1, \dots, x_n)\}.$$

**Definition 2.1.** Let  $B, \ell$  be positive integers. We say a vector  $x = (x_1, \dots, x_\ell) \in \mathbb{Z}^\ell$  is  $B$ -bounded if  $|x_i| \leq B$  for all  $i \in [\ell]$ . A distribution  $\mathcal{D}$  over  $\mathbb{Z}^\ell$  is  $B$ -bounded if  $\text{Support}(\mathcal{D}) \subseteq [-B, B]^\ell$ . For a vector  $\mathbf{B} = (B_1, \dots, B_\ell) \in \mathbb{Z}^\ell$ , we say  $x$  is  $\mathbf{B}$ -bounded if  $|x_i| \leq B_i$  for all  $i \in [\ell]$ , and  $\mathcal{D}$  is  $\mathbf{B}$ -bounded if all elements in the support of  $\mathcal{D}$  are  $\mathbf{B}$ -bounded.

**Definition 2.2** (Statistical Distance). Let  $X$  and  $X'$  be random variables over a discrete set  $\mathcal{X}$ . The *statistical distance* between  $X$  and  $X'$  is defined as

$$\delta(X, X') := \frac{1}{2} \sum_{x \in \mathcal{X}} |\Pr[X = x] - \Pr[X' = x]|.$$

The *min-entropy* of a random variable  $X$  is defined as

$$H_\infty(X) := -\log\left(\max_x \Pr[X = x]\right).$$

We further define the *conditional min-entropy* of  $X$  given  $Z$  following Dodis et al. [DORS08] as

$$H_\infty(X | Z) := -\log\left(\mathbb{E}_{z \leftarrow Z} \left[ \max_x \Pr[X = x | Z = z] \right]\right).$$

We denote by PPT probabilistic polynomial time Turing machines. The term *negligible* is used for denoting functions that are (asymptotically) smaller than any inverse polynomial. More precisely, a function  $\nu$  from  $\mathbb{N}$  to reals is called *negligible* if for every constant  $c > 0$  and all sufficiently large  $n$ ,  $\nu(n) < n^{-c}$ .

**Definition 2.3** ( $\mu$ -indistinguishability). Let  $\mu: \mathbb{N} \rightarrow [0, 1]$  be a function. A pair of distribution ensembles  $\{X_\lambda\}_{\lambda \in \mathbb{N}}$ ,  $\{Y_\lambda\}_{\lambda \in \mathbb{N}}$  are  $\mu$ -indistinguishable if for every family of polynomial-sized distinguishers  $\{D_\lambda\}_{\lambda \in \mathbb{N}}$ , and every sufficiently large security parameter  $\lambda \in \mathbb{N}$ ,

$$|\Pr[x \leftarrow X_\lambda : D_\lambda(x) = 1] - \Pr[y \leftarrow Y_\lambda : D_\lambda(y) = 1]| \leq O(\mu(\lambda)).$$

## 2.2 Learning with Errors

We next state the decisional learning with errors (LWE) assumption, which was introduced by Regev [Reg05].

**Definition 2.4.** Let  $n = n(\lambda)$ ,  $m = m(\lambda)$ , and  $q = q(\lambda)$  be integers and let  $\chi = \chi(\lambda)$  be a distribution over  $\mathbb{Z}_q(\lambda)$  for  $\lambda \in \mathbb{N}$ . Then, the  $\text{LWE}_{n,m,q,\chi}$  assumption with  $\mu$ -indistinguishability is that the following distributions are  $\mu$ -indistinguishable:

$$\left\{ \mathbf{A} \leftarrow \mathbb{Z}_q^{m \times n}; \mathbf{s} \leftarrow \mathbb{Z}_q^n; \mathbf{e} \leftarrow \chi^m : (\mathbf{A}, \mathbf{A} \cdot \mathbf{s} + \mathbf{e}) \right\}_{\lambda \in \mathbb{N}},$$

$$\left\{ \mathbf{A} \leftarrow \mathbb{Z}_q^{m \times n}; \mathbf{u} \leftarrow \mathbb{Z}_q^m : (\mathbf{A}, \mathbf{u}) \right\}_{\lambda \in \mathbb{N}}.$$

It has been shown that this assumptions holds when  $\chi$  is a discrete Gaussian distribution if certain worst-case lattice problems are hard [Reg05, Pei09].

We further define LWE with weak and leaky secrets, as introduced in the full version of [AKPW13].

**Definition 2.5** (LWE with Weak and Leaky Secrets). Let  $n = n(\lambda)$ ,  $m = m(\lambda)$ ,  $q = q(\lambda)$ , and  $\gamma = \gamma(\lambda) \in (0, q/2) \cap \mathbb{Z}$  be integers, let  $k = k(\lambda)$  be a real, and let  $\chi = \chi(\lambda)$  be a distribution over  $\mathbb{Z}_q(\lambda)$  for  $\lambda \in \mathbb{N}$ . Then, the  $\text{LWE}_{n,m,q,\chi}^{\text{WL}(\gamma,k)}$  assumption with  $\mu$ -indistinguishability states that for all efficiently samplable correlated random variables  $(\mathbf{s}, \text{aux})$ , where the support of  $\mathbf{s}$  is  $[-\gamma, \gamma]^n \cap \mathbb{Z}^n$  and  $H_\infty(\mathbf{s} \mid \text{aux}) \geq k$ , the following distributions are  $\mu$ -indistinguishable

$$(\text{aux}, \mathbf{A}, \mathbf{A} \cdot \mathbf{s} + \mathbf{e}), \quad (\text{aux}, \mathbf{A}, \mathbf{u}),$$

where  $\mathbf{A} \leftarrow \mathbb{Z}_q^{m \times n}$ ,  $\mathbf{e} \leftarrow \chi^m$ , and  $\mathbf{u} \leftarrow \mathbb{Z}_q^m$  are sampled independently of  $(\mathbf{s}, \text{aux})$ .

### 2.3 Asymmetric Bilinear Maps and the SXDH Assumption

We first define bilinear maps and then the symmetric external Diffie-Hellman (SXDH) assumption.

**Definition 2.6** (Bilinear maps). A bilinear map generator  $\mathcal{G}$  on input a security parameter  $1^\lambda$ , outputs  $(p, G_1, G_2, G_3, \text{pair})$ , where  $G_1$ ,  $G_2$ , and  $G_3$  are (descriptions of) cyclic groups of order  $p$  (which can be prime or composite). The groups  $G_1$  and  $G_2$  are called source groups, and  $G_3$  is called target group. We assume the descriptions of the source groups  $G_1$  and  $G_2$  contain generators  $g_1$  and  $g_2$ , respectively. Moreover,  $\text{pair}$  satisfies the following:

- *Admissibility*: We have that  $\text{pair}: G_1 \times G_2 \rightarrow G_3$  is an efficiently computable function such that  $g_3 := \text{pair}(g_1, g_2)$  generates  $G_3$ .
- *Bilinearity*: For all  $a_1, a_2 \in \mathbb{Z}_p$ ,  $\text{pair}(g_1^{a_1}, g_2^{a_2}) = \text{pair}(g_1, g_2)^{a_1 a_2} = g_3^{a_1 a_2}$ .

In this work, we will use the bracket notation  $[x]_l = g_l^x$  to represent elements in group  $G_l$ .

We next state the SXDH assumption, which says that the DDH assumption holds in all source groups.

**Definition 2.7** (SXDH assumption). Let  $\mathcal{G}$  be a bilinear group generator. The SXDH assumption is that the following two distributions are  $\mu$ -indistinguishable for all  $i \in \{1, 2\}$ :

$$\begin{aligned} & \{ \text{params} = (p, G_1, G_2, G_3, \text{pair}) \leftarrow \mathcal{G}(1^\lambda), a, b \leftarrow \mathbb{Z}_p : (\text{params}, [a]_i, [b]_i, [ab]_i) \}_{\lambda \in \mathbb{N}}, \\ & \{ \text{params} = (p, G_1, G_2, G_3, \text{pair}) \leftarrow \mathcal{G}(1^\lambda), a, b, c \leftarrow \mathbb{Z}_p : (\text{params}, [a]_i, [b]_i, [c]_i) \}_{\lambda \in \mathbb{N}}. \end{aligned}$$

### 2.4 Pseudorandom Generators and Pseudorandom Functions

We review the notion of a pseudorandom generator (PRG) family and its locality.

**Definition 2.8** (Family of Pseudorandom Generators (PRGs)). Let  $n$  and  $m$  be polynomials. A family of  $(n, m)$ -PRGs is an ensemble of distributions  $\text{PRG} = \{\text{PRG}_\lambda\}_{\lambda \in \mathbb{N}}$  satisfying the following properties:

**Syntax**: For every  $\lambda \in \mathbb{N}$ , every PRG in the support of  $\text{PRG}_\lambda$  defines a function  $\{0, 1\}^{n(\lambda)} \rightarrow \{0, 1\}^{m(\lambda)}$ , for which we also write PRG.

**Efficiency**: There is a uniform Turing machine  $M$  satisfying that for every  $\lambda \in \mathbb{N}$ , every PRG in the support of  $\text{PRG}_\lambda$ , and for every  $x \in \{0, 1\}^{n(\lambda)}$ ,  $M(\text{PRG}, x)$  runs in time  $\text{poly}(\lambda)$  and we have  $M(\text{PRG}, x) = \text{PRG}(x)$ .

**$\mu$ -Indistinguishability:** The following ensembles are  $\mu$ -indistinguishable:

$$\left\{ \text{PRG} \leftarrow \text{PRG}_\lambda; s \leftarrow \{0, 1\}^{n(\lambda)} : (\text{PRG}, \text{PRG}(s)) \right\}_\lambda,$$

$$\left\{ \text{PRG} \leftarrow \text{PRG}_\lambda; r \leftarrow \{0, 1\}^{m(\lambda)} : (\text{PRG}, r) \right\}_\lambda.$$

**Definition 2.9** (Locality of PRGs). Let  $n$ ,  $m$ , and  $\ell$  be polynomials. We say a family of  $(n, m)$ -PRGs  $\text{PRG}$  has locality  $\ell$  if for every  $\lambda$  and for every  $\text{PRG}$  in the support of  $\text{PRG}_\lambda$ , every output bit of  $\text{PRG}$  depends on at most  $\ell(\lambda)$  input bits.

We next define pseudorandom function (PRF) families.

**Definition 2.10.** For  $\lambda \in \mathbb{N}$ , let  $\mathcal{K} = \mathcal{K}(\lambda)$ ,  $X = X(\lambda)$ , and  $Y = Y(\lambda)$  be finite sets, and let  $\text{PRF} = \text{PRF}_\lambda: \mathcal{K} \times X \rightarrow Y$  be an efficiently computable function. We say  $(\text{PRF}_\lambda)_{\lambda \in \mathbb{N}}$  is a pseudorandom function family with  $\mu$ -indistinguishability if for all PPT algorithms  $A$  with access to an oracle, which is either  $\text{PRF}(K, \cdot)$  for  $K \leftarrow \mathcal{K}$  or a truly uniform function  $X \rightarrow Y$ ,

$$\left| \Pr \left[ K \leftarrow \mathcal{K} : A^{\text{PRF}(K, \cdot)}(1^\lambda) = 1 \right] - \Pr \left[ U \leftarrow (X \rightarrow Y) : A^{U(\cdot)}(1^\lambda) = 1 \right] \right| \leq O(\mu(\lambda)).$$

## 2.5 Symmetric Encryption

Now, we give definitions for basic symmetric encryption schemes and IND-CPA-security.

**Definition 2.11** (Symmetric encryption). A symmetric encryption scheme  $\text{Sym}$  consists of the following PPT algorithms:

- *Key Generation:*  $\text{Sym.KeyGen}(1^\lambda)$  on input a security parameter  $1^\lambda$ , outputs a key  $K$ .
- *Encryption:*  $\text{Sym.Enc}(K, m)$  on input a key  $K$  and a message  $m$ , outputs a ciphertext  $c$ .
- *Decryption:*  $\text{Sym.Dec}(K, c)$  on input a key  $K$  and a ciphertext  $c$ , outputs a message  $m'$ .

For correctness, we require for all  $\lambda$  and for all messages  $m$ , for  $K \leftarrow \text{Sym.KeyGen}(1^\lambda)$ , and  $c \leftarrow \text{Sym.Enc}(K, m)$ , that  $\text{Sym.Dec}(K, c) = m$  with probability 1.

**Definition 2.12** (IND-CPA-security). A symmetric encryption scheme  $\text{Sym} = (\text{Sym.KeyGen}, \text{Sym.Enc}, \text{Sym.Dec})$  is  $\mu$ -IND-CPA-secure if for every PPT adversary  $A$  and for every sufficiently large  $\lambda$ , the advantage of  $A$  in the following game is bounded by  $O(\mu(\lambda))$ :

- The challenger runs  $K \leftarrow \text{Sym.KeyGen}(1^\lambda)$ .
- The adversary  $A$  with access to an encryption oracle  $\text{Sym.Enc}(K, \cdot)$  chooses a pair of messages  $m_0, m_1$  of equal length and sends them to the challenger.
- The challenger samples a bit  $b \leftarrow \{0, 1\}$ , computes  $c \leftarrow \text{Sym.Enc}(K, m_b)$ , and sends  $c$  to  $A$ .
- The adversary  $A$  again has access to an encryption oracle  $\text{Sym.Enc}(K, \cdot)$ , and finally outputs a bit  $b'$ .

The advantage of  $A$  is defined as

$$\text{Adv}_A^{\text{Sym}} := |2 \cdot \Pr[b' = b] - 1|.$$

## 2.6 Indistinguishability Obfuscation

We recall the notion of indistinguishability obfuscation for a class of circuits defined by [BGI<sup>+</sup>01].

**Definition 2.13** (Indistinguishability Obfuscator ( $i\mathcal{O}$ ) for a circuit class). A uniform PPT machine  $i\mathcal{O}$  is an indistinguishability obfuscator for a class of circuits  $\{\mathcal{C}_\lambda\}_{\lambda \in \mathbb{N}}$  if the following conditions are satisfied:

**Correctness:** For all security parameters  $\lambda \in \mathbb{N}$ , for every  $C \in \mathcal{C}_\lambda$ , and every input  $x$ , we have

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

where the probability is taken over the coin-tosses of the obfuscator  $i\mathcal{O}$ .

**$\mu$ -Indistinguishability:** For every ensemble of pairs of circuits  $\{C_{0,\lambda}, C_{1,\lambda}\}_{\lambda \in \mathbb{N}}$  satisfying that  $C_{b,\lambda} \in \mathcal{C}_\lambda$ ,  $|C_{0,\lambda}| = |C_{1,\lambda}|$ , and  $C_{0,\lambda}(x) = C_{1,\lambda}(x)$  for every  $x$ , the following ensembles of distributions are  $\mu$ -indistinguishable:

$$\begin{aligned} & \{C_{1,\lambda}, C_{2,\lambda}, i\mathcal{O}(1^\lambda, C_{1,\lambda})\}_{\lambda \in \mathbb{N}}, \\ & \{C_{1,\lambda}, C_{2,\lambda}, i\mathcal{O}(1^\lambda, C_{2,\lambda})\}_{\lambda \in \mathbb{N}}. \end{aligned}$$

**Definition 2.14** (IO for P/poly). A uniform PPT machine  $i\mathcal{O}_{\text{P/poly}}(\star, \star)$  is an indistinguishability obfuscator for P/poly if it is an indistinguishability obfuscator for the class  $\{\mathcal{C}_\lambda\}_{\lambda \in \mathbb{N}}$  of circuits of size at most  $\lambda$ .

## 2.7 Randomized Encodings

In this section, we recall the traditional definition of randomized encodings with simulation security [IK02, AIK06].

**Definition 2.15** (Randomized encoding scheme for circuits). A randomized encoding scheme RE consists of the following two PPT algorithms:

- $\hat{C}_x \leftarrow \text{REnc}(1^\lambda, C, x)$ : On input a security parameter  $1^\lambda$ , circuit  $C$ , and input  $x$ , REnc generates an encoding  $\hat{C}_x$ .
- $y = \text{REval}(\hat{C}_x)$ : On input  $\hat{C}_x$  produced by REnc, REval outputs  $y$ .

**Correctness:** For all security parameters  $\lambda \in \mathbb{N}$ , circuits  $C$ , and inputs  $x$ , it holds that

$$\Pr[\hat{C}_x \leftarrow \text{REnc}(1^\lambda, C, x) : \text{Eval}(\hat{C}_x) = C(x)] = 1.$$

**$\mu$ -Simulation Security:** There exists a PPT algorithm RSim such that for every ensemble  $\{C_\lambda, x_\lambda\}_\lambda$  where  $|C_\lambda|, |x_\lambda| \leq \text{poly}(\lambda)$ , the following ensembles are  $\mu$ -indistinguishable for all  $\lambda \in \mathbb{N}$ :

$$\begin{aligned} & \left\{ \hat{C}_x \leftarrow \text{REnc}(1^\lambda, C, x) : \hat{C}_x \right\}_{\lambda \in \mathbb{N}}, \\ & \left\{ \hat{C}_x \leftarrow \text{RSim}(1^\lambda, C(x), 1^{|C|}, 1^{|x|}) : \hat{C}_x \right\}_{\lambda \in \mathbb{N}}, \end{aligned}$$

where  $C = C_\lambda$  and  $x = x_\lambda$ .

Furthermore, let  $\mathcal{C}$  be a complexity class. We say that the randomized encoding scheme RE is in  $\mathcal{C}$  if the encoding algorithm REnc can be implemented in that complexity class.

## 2.8 Functional Encryption

### 2.8.1 Secret-Key Functional Encryption

We provide the definition of a secret-key functional encryption (FE) scheme, which is an adaptation of the public-key definition that originally appeared in [BSW11, O'N10]. We further define indistinguishability-based and simulation-based selective security, for setting in which a single function key is released, though the function may have multiple output elements. This is because most part of the paper uses only 1-key FE. Nevertheless, all definitions can be easily extended to the multi-key setting.

**Definition 2.16** (Secret-key FE). Let  $\mathcal{X} = \{\mathcal{X}_\lambda\}_{\lambda \in \mathbb{N}}$  and  $\mathcal{Y} = \{\mathcal{Y}_\lambda\}_{\lambda \in \mathbb{N}}$  be ensembles of sets. Let  $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda \in \mathbb{N}}$ , where every function in the set  $\mathcal{F}_\lambda$  maps inputs in  $\mathcal{X}_\lambda$  to outputs in  $\mathcal{Y}_\lambda$ . A secret-key functional encryption scheme FE for  $\{\mathcal{F}_\lambda\}_{\lambda \in \mathbb{N}}$  consists of four PPT algorithms (FE.Setup, FE.KeyGen, FE.Enc, FE.Dec) such that

- *Setup*: FE.Setup( $1^\lambda$ ) is an algorithm that on input a security parameter, outputs a master secret key  $\text{msk}$ .
- *Key Generation*: FE.KeyGen( $\text{msk}, f$ ) on input the master secret key  $\text{msk}$  and the description of a function  $f \in \mathcal{F}_\lambda$ , outputs a secret key  $\text{sk}_f$ .
- *Encryption*: FE.Enc( $\text{msk}, x$ ) on input the master secret key  $\text{msk}$  and a message  $x \in \mathcal{X}_\lambda$ , outputs an encryption  $\text{ct}$  of  $x$ .
- *Decryption*: FE.Dec( $\text{sk}, \text{ct}$ ) on input the secret key associated with  $f$  and an encryption of  $x$ , outputs  $y \in \mathcal{Y}_\lambda$ .

**Correctness.** We define perfect correctness here. For every  $\lambda, f \in \mathcal{F}_\lambda, x \in \mathcal{X}_\lambda$ , it holds that,

$$\Pr \left[ \begin{array}{l} \text{msk} \leftarrow \text{FE.Setup}(1^\lambda) \\ \text{ct} \leftarrow \text{FE.Enc}(\text{msk}, x) \\ \text{sk} \leftarrow \text{FE.KeyGen}(\text{msk}, f) \end{array} : f(x) = \text{FE.Dec}(\text{sk}, \text{ct}) \right] = 1.$$

**Indistinguishability security.** Indistinguishability security of functional encryption requires that no adversary can distinguish the FE encryptions of one sequence of inputs  $x_1^0, \dots, x_t^0$  from that of another  $x_1^1, \dots, x_t^1$ , if the adversary only obtains secret keys for functions that yield the same outputs on  $x_i^0$  and  $x_i^1$  for every  $i$ .

**Definition 2.17** (1-key Sel-Ind-security). A secret-key functional encryption scheme FE = (FE.Setup, FE.KeyGen, FE.Enc, FE.Dec) for  $\{\mathcal{F}_\lambda\}_{\lambda \in \mathbb{N}}$  is *1-key  $\mu$ -Sel-Ind-secure*, if for every ensemble of functions  $\{f_\lambda\}_{\lambda \in \mathbb{N}}$  where  $f_\lambda \in \mathcal{F}_\lambda$ , every polynomial  $t$ , and every ensemble of sequences of pairs of inputs  $\{x_{i,\lambda}^0, x_{i,\lambda}^1\}_{\lambda \in \mathbb{N}, i \in [t(\lambda)]}$  where  $x_{i,\lambda}^0, x_{i,\lambda}^1 \in \mathcal{X}_\lambda$  and  $f_\lambda(x_{i,\lambda}^0) = f_\lambda(x_{i,\lambda}^1)$ , the following distributions for  $b = 0$  and  $b = 1$  are  $\mu$ -indistinguishable:

$$\left\{ \begin{array}{l} \text{msk} \leftarrow \text{FE.Setup}(1^\lambda) \\ \text{sk} \leftarrow \text{FE.KeyGen}(\text{msk}, f_\lambda) \\ \left\{ \text{ct}_i \leftarrow \text{FE.Enc}(\text{msk}, x_{i,\lambda}^b) \right\}_{i \in [t(\lambda)]} \end{array} : \text{sk}, \left\{ \text{ct}_i \right\}_{i \in [t(\lambda)]} \right\}_{\lambda \in \mathbb{N}}.$$

Note that our notion of fully-selective security is weaker than the notion of selective security in some papers in the literature (e.g., [GKP<sup>+</sup>13, ABSV15]), which only requires the adversaries to choose challenge inputs statically, but allows the adversaries to choose challenge function inputs adaptively. Intuitively, the notion of fully-selective security is sufficient for applications that are non-interactive, for instance, building IO from FE as in [AJ15, BV15].

**Simulation security.** We will also consider 1-key simulation-based security, which essentially requires that the secret key for a function  $f$  and the ciphertext for an input  $x$  can be simulated by a simulator receiving only the output  $f(x)$ . Our definition is slightly stronger. Since adversaries do not have the capability of generating ciphertexts in the secret key setting, the definition below allows the adversaries to see multiple ciphertexts, and the simulator is required to simulate all ciphertexts using the output for one input, and the other actual inputs.

**Definition 2.18** (1-key Sel-Sim-security). A secret-key functional encryption scheme  $\text{FE} = (\text{FE.Setup}, \text{FE.KeyGen}, \text{FE.Enc}, \text{FE.Dec})$  for  $\{\mathcal{F}_\lambda\}_{\lambda \in \mathbb{N}}$  is *1-key  $\mu$ -Sel-Sim-secure*, if there is a PPT universal simulator  $\text{Sim}$  such that, for every ensemble of functions  $\{f_\lambda\}_{\lambda \in \mathbb{N}}$  where  $f_\lambda \in \mathcal{F}_\lambda$ , every ensemble of inputs  $\{x_\lambda^*\}_{\lambda \in \mathbb{N}}$ , every polynomial  $t$ , and every ensemble of sequences of inputs  $\{x_{i,\lambda}\}_{\lambda \in \mathbb{N}, i \in [t(\lambda)]}$ , where  $x_\lambda^*, x_{i,\lambda} \in \mathcal{X}_\lambda$ , the following distributions are  $\mu$ -indistinguishable:

$$\left\{ \begin{array}{l} \text{msk} \leftarrow \text{FE.Setup}(1^\lambda) \\ \text{sk} \leftarrow \text{FE.KeyGen}(\text{msk}, f_\lambda) \\ \text{ct}^* \leftarrow \text{FE.Enc}(\text{msk}, x_\lambda^*) \\ \{\text{ct}_i \leftarrow \text{FE.Enc}(\text{msk}, x_{i,\lambda})\}_{i \in [t(\lambda)]} \end{array} : \text{sk}, \text{ct}^*, \{\text{ct}_i\}_{i \in [t(\lambda)]} \right\}_{\lambda \in \mathbb{N}},$$

$$\left\{ \text{Sim} \left( f_\lambda, f_\lambda(x_\lambda^*), \{x_{i,\lambda}\}_{i \in [t(\lambda)]} \right) \right\}_{\lambda \in \mathbb{N}}.$$

**Function Hiding.** In the literature, there is also a stronger notion of security for secret key FE, called *function hiding*<sup>5</sup>, which roughly speaking requires the scheme to hide both information of the encrypted inputs, as well as, the functions encoded in secret keys. We now define the notion of function hiding in the fully selective and multi-key setting.

**Definition 2.19** (Function hiding). A secret-key FE scheme  $\text{FE} = (\text{FE.Setup}, \text{FE.KeyGen}, \text{FE.Enc}, \text{FE.Dec})$  for  $\{\mathcal{F}_\lambda\}_{\lambda \in \mathbb{N}}$  is  $\mu$ -Sel-FH, if for every polynomial  $t$ , every  $\{f_{i,\lambda}^0, f_{i,\lambda}^1\}_{i \in [t(\lambda)], \lambda \in \mathbb{N}}$ , and every  $\{x_{i,\lambda}^0, x_{i,\lambda}^1\}_{\lambda \in \mathbb{N}, i \in [t(\lambda)]}$ , where  $f_{i,\lambda}^0, f_{i,\lambda}^1 \in \mathcal{F}_\lambda$ ,  $x_{i,\lambda}^0, x_{i,\lambda}^1 \in \mathcal{X}_\lambda$  and  $f_\lambda^0(x_{i,\lambda}^0) = f_\lambda^1(x_{i,\lambda}^1)$ , the following distributions for  $b = 0$  and  $b = 1$  are  $\mu$ -indistinguishable:

$$\left\{ \begin{array}{l} \text{msk} \leftarrow \text{FE.Setup}(1^\lambda) \\ \left\{ \text{sk}_i \leftarrow \text{FE.KeyGen}(\text{msk}, f_{i,\lambda}^b) \right\}_{i \in [t(\lambda)]} \\ \left\{ \text{ct}_i \leftarrow \text{FE.Enc}(\text{msk}, x_{i,\lambda}^b) \right\}_{i \in [t(\lambda)]} \end{array} : \left\{ \text{sk} \right\}_{i \in [t(\lambda)]}, \left\{ \text{ct}_i \right\}_{i \in [t(\lambda)]} \right\}_{\lambda \in \mathbb{N}}.$$

### 2.8.2 FE for P/poly, NC<sup>1</sup>, and Inner Products, and Compactness

**Definition 2.20** (FE schemes for families of function classes). Let  $\{\mathcal{F}^I\}_{I \in \mathcal{I}}$  be a family of function classes. We say that  $\{\text{FE}^I\}_{I \in \mathcal{I}}$  is a family of (1-key) FE schemes for  $\{\mathcal{F}^I\}_{I \in \mathcal{I}}$  with

<sup>5</sup>Public key FE cannot be function hiding.

$\mu$ -Sel-Ind-security (or  $\mu$ -Sel-Sim-security) if for every function class  $\mathcal{F}^I = \{\mathcal{F}_\lambda^I\}_{\lambda \in \mathbb{N}}$ ,  $\text{FE}^I$  is a (1-key) FE scheme for  $\mathcal{F}^I$  with  $\mu$ -Sel-Ind-security (or  $\mu$ -Sel-Sim-security).

We further define the following special cases:

- **FE for P/poly** is a family of FE schemes for  $\{\text{P/poly}^{N,S}\}_{N \in \mathcal{N}, S \in \mathcal{S}}$ , where  $\mathcal{N}, \mathcal{S}$  are the sets of all polynomials and  $\text{P/poly}^{N,S}$  is the class of functions that can be computed by circuits with  $N(\lambda)$ -bit inputs and size  $S(\lambda)$ .
- **FE for  $\text{NC}^1$**  is a family of FE schemes for  $\{\text{NC}^{1,N,S,\text{Dep}}\}_{N \in \mathcal{N}, S \in \mathcal{S}, \text{Dep} \in \mathcal{D}}$  as defined above, but only for circuits with logarithmic depth  $\text{Dep}(\lambda) = O(\log(\lambda))$ , where  $\mathcal{D}$  is the set of logarithmic functions.
- **FE for inner products in  $\mathcal{R}$**  is a family of FE schemes for  $\mathbb{F} = \{\mathcal{F}^N\}$  where  $\mathcal{F}^N$  is the set of functions of form  $f_{\mathbf{v}}(\mathbf{x}) = \langle \mathbf{v}, \mathbf{x} \rangle$  that compute the inner product between a fixed vector  $\mathbf{v}$  and an input vector  $\mathbf{x}$  in  $\mathcal{R}_\lambda^{N(\lambda)}$ . Such a family of schemes is also called Inner Product Functional Encryption (IPE) in  $\mathcal{R}$ .

**Compactness.** Let  $\{\text{FE}^{N,S,Z}\}_{N \in \mathcal{N}, N \in \mathcal{S}, Z \in \mathcal{Z}}$  be a family of FE schemes for  $\{\mathcal{F}^{N,S,Z}\}$ , where  $\mathcal{F}_\lambda^{N,S,Z}$  contains a subset of functions with input lengths bounded by  $N(\lambda)$  (i.e.,  $\mathcal{X}_\lambda = \Delta^{\leq N(\lambda)}$  for some alphabet  $\Delta$ ) and size bounded by  $S(\lambda)$  (e.g.,  $\text{P/poly}^{N,S}$  or  $\text{NC}^{1,N,S,\text{Dep}}$ ). According to the above definition, algorithms in the FE schemes could run in time polynomial in the parameters  $N$  and  $S$ . Stronger efficiency requirements have been considered; in particular, the works of [AJ15, BV15] defined compact FE schemes, which requires the *encryption time* to be independent of, or only mildly dependent on, the circuit size  $S$  of the functions. Here, we consider a relaxation that only requires the *ciphertext size* to be independent of, or only mildly dependent on  $S$ , whereas the encryption time can still be polynomial in  $S$ . See Remark 5.3 for reasons for using this more relaxed notion of compactness.

**Definition 2.21** (Ciphertext-Compactness of FE schemes). Let  $\{\text{FE}^{N,S,Z}\}_{N \in \mathcal{N}, N \in \mathcal{S}, Z \in \mathcal{Z}}$  be a family of FE schemes for  $\{\mathcal{F}^{N,S,Z}\}_{N \in \mathcal{N}, N \in \mathcal{S}, Z \in \mathcal{Z}}$ , where functions in  $\mathcal{F}_\lambda^{N,S,Z}$  have input length bounded by  $N(\lambda)$  and size bounded by  $S(\lambda)$ .

**Ciphertext-Compactness:** We say that  $\{F^{N,S,Z}\}$  is ciphertext-compact if for every  $Z \in \mathcal{Z}$ , there exists a polynomial  $p$  such that, for all polynomials  $N$  and  $S$ , ciphertexts of  $\text{FE}^{N,S,Z}$  have size  $p(\lambda, N(\lambda), \log S(\lambda))$ .

**$(1 - \varepsilon)$ -Sublinear Ciphertext-Compactness:** We say that  $\{F^{N,S,Z}\}$  is  $(1 - \varepsilon)$ -sublinearly ciphertext-compact if for every  $Z \in \mathcal{Z}$ , there exists a polynomial  $p$  such that, for all polynomials  $N$  and  $S$ , ciphertexts of  $\text{FE}^{N,S,Z}$  have size  $p(\lambda, N(\lambda)) \cdot S(\lambda)^{1-\varepsilon}$ .

In the rest of the paper, by compactness, we mean by default ciphertext-compactness.

## 2.9 (Fully) Homomorphic Encryption

We give a definition of secret key fully homomorphic encryption following [Gen09a, Gen09b].

**Definition 2.22** ((Fully) Homomorphic Encryption). A fully homomorphic secret-key encryption (FHE) scheme FHE consists of the following four PPT algorithms:

**Key generation:** The algorithm  $\text{FHE.KeyGen}$  on input a security parameter  $1^\lambda$ , outputs a key  $s$ .

**Encryption:** The algorithm  $\text{FHE.Enc}$  on input a key  $s$  and a message  $x$ , outputs a ciphertext  $\text{ct}$ .

**Evaluation:** The algorithm  $\text{FHE.Eval}$  on input a circuit  $C$  and a tuple of ciphertexts  $(\text{ct}_1, \dots, \text{ct}_t)$ , outputs a ciphertext  $\text{ct}'$ .

**Decryption:** The algorithm  $\text{FHE.Dec}$  on input a key  $s$  and a ciphertext  $\text{ct}$ , outputs a message  $y$ .

An FHE scheme is required to have to following two properties:

**Correctness:** For all  $s$  output by  $\text{FHE.KeyGen}(1^\lambda)$ , all circuits  $C$ , all messages  $x_1, \dots, x_t$ , and all ciphertexts  $\text{ct}_1, \dots, \text{ct}_t$  output by  $\text{FHE.Enc}(s, x_1), \dots, \text{FHE.Enc}(s, x_t)$ , we have

$$\Pr[\text{ct}' \leftarrow \text{FHE.Eval}(C, (\text{ct}_1, \dots, \text{ct}_t)) : \text{FHE.Dec}(s, \text{ct}') = C(x_1, \dots, x_t)] = 1 - \text{negl}(\lambda).$$

**Compactness:** There is a polynomial  $p$  such that for all  $s$  output by  $\text{FHE.KeyGen}(1^\lambda)$ , all circuits  $C$ , all messages  $x_1, \dots, x_t$ , all ciphertexts  $\text{ct}_1, \dots, \text{ct}_t$  output by  $\text{FHE.Enc}(s, x_1), \dots, \text{FHE.Enc}(s, x_t)$ , and all  $\text{ct}'$  output by  $\text{FHE.Eval}(C, (\text{ct}_1, \dots, \text{ct}_t))$ , we have that  $|\text{ct}'| \leq p(\lambda)$  and  $\text{FHE.Dec}(s, \text{ct}')$  runs in time bounded by  $p(\lambda)$  (independently of  $C$ ).

If the evaluation only allows circuits from a certain class (and correctness and compactness holds for such circuits), we call the scheme homomorphic for that class (instead of fully homomorphic).

**Definition 2.23** (Leveled Homomorphic Encryption). A family of homomorphic encryption schemes  $\{\text{HE}^{(d)}\}_{d \in \mathbb{N}}$ , where for all  $d \in \mathbb{N}$ ,  $\text{HE}^{(d)}$  is homomorphic for all circuits of depth at most  $d$ , is called leveled fully homomorphic if all  $\text{HE}^{(d)}$  use the same decryption circuit, and the computational complexity of all algorithms in  $\text{HE}^{(d)}$  is polynomial in  $\lambda, d$ , and (for the evaluation algorithm) the size of the circuit.

The definition of IND-CPA-security for (fully) homomorphic encryption is identical to the definition for ordinary secret-key encryption:

**Definition 2.24** (IND-CPA-Security). Let  $\text{FHE} = (\text{FHE.KeyGen}, \text{FHE.Enc}, \text{FHE.Eval}, \text{FHE.Dec})$  be an FHE scheme. The scheme  $\text{FHE}$  is  $\mu$ -IND-CPA-secure if for every PPT adversary  $A$  and for every sufficiently large  $\lambda$ , the advantage of  $A$  in the following game is bounded by  $O(\mu(\lambda))$ :

- The challenger runs  $s \leftarrow \text{FHE.KeyGen}(1^\lambda)$ .
- The adversary  $A$  with access to an encryption oracle  $\text{FHE.Enc}(s, \cdot)$  chooses a pair of messages  $x_0, x_1$  of equal length and sends them to the challenger.
- The challenger samples a bit  $b \leftarrow \{0, 1\}$ , computes  $\text{ct} \leftarrow \text{FHE.Enc}(s, x_b)$ , and sends  $\text{ct}$  to  $A$ .
- The adversary  $A$  again has access to an encryption oracle  $\text{FHE.Enc}(s, \cdot)$ , and finally outputs a bit  $b'$ .

The advantage of  $A$  is defined as

$$\text{Adv}_A^{\text{FHE}} := |2 \cdot \Pr[b' = b] - 1|.$$

### 2.9.1 Threshold Multi-Key FHE

We next give definitions for multi-key FHE [LATV12] and threshold multi-key FHE following [MW16].

**Definition 2.25** (Multi-Key (Leveled) FHE). A multi-key (leveled) FHE scheme consists of the following PPT algorithms:

- **MFHE.Setup** on input a security parameter  $1^\lambda$  and circuit depth  $1^d$ , outputs the system parameters  $\text{params}$ .
- **MFHE.KeyGen** on input the system parameters  $\text{params}$ , outputs a public key  $\text{pk}$  and a secret key  $\text{sk}$ .
- **MFHE.Enc** on input  $\text{pk}$  and a message  $x$ , outputs a ciphertext  $\text{ct}$ .
- **MFHE.Expand** on input a sequence of public keys  $\text{pk}_1, \dots, \text{pk}_N$ , an index  $i \in [N]$ , and a fresh ciphertext  $c$  under the  $i$ th key  $\text{pk}_i$ , outputs an “expanded” ciphertext  $\widehat{\text{ct}}$ .
- **MFHE.Eval** on input  $\text{params}$ , a boolean circuit  $C$  of depth at most  $d$ , and expanded ciphertexts  $\widehat{\text{ct}}_1, \dots, \widehat{\text{ct}}_\ell$ , outputs an evaluated ciphertext  $\widehat{\text{ct}}$ .
- **MFHE.Dec** on input  $\text{params}$ , a sequence of secret keys  $\text{sk}_1, \dots, \text{sk}_N$ , and a ciphertext  $\widehat{\text{ct}}$ , outputs a message.

We require the following properties:

**Correctness and Compactness:** Let  $\text{params} \leftarrow \text{MFHE.Setup}(1^\lambda, 1^d)$ , for  $i \in \{1, \dots, N\}$ , let  $(\text{pk}_i, \text{sk}_i) \leftarrow \text{MFHE.KeyGen}(\text{params})$ , and let  $x_1, \dots, x_\ell$  be messages. Moreover, let  $I_1 \in [N], \dots, I_\ell \in [N]$ ,  $\text{ct}_i \leftarrow \text{MFHE.Enc}(\text{pk}_{I_i}, x_i)$ , and let  $\widehat{\text{ct}}_i \leftarrow \text{MFHE.Expand}((\text{pk}_1, \dots, \text{pk}_N), I_i, \text{ct}_i)$  for  $i \in [\ell]$ . Let  $C$  be a circuit of depth at most  $d$  and let  $\widehat{\text{ct}} := \text{MFHE.Eval}(C, (\widehat{\text{ct}}_1, \dots, \widehat{\text{ct}}_\ell))$ . Then the following holds:

**Correctness of Expansion:**  $\text{MFHE.Dec}(\text{params}, (\text{sk}_1, \dots, \text{sk}_N), \widehat{\text{ct}}_i) = x_i$  for all  $i \in [\ell]$ .

**Correctness of Evaluation:**  $\text{MFHE.Dec}(\text{params}, (\text{sk}_1, \dots, \text{sk}_N), \widehat{\text{ct}}) = C(x_1, \dots, x_\ell)$ .

**Compactness:**  $|\widehat{\text{ct}}|$  is polynomial in  $\lambda$ ,  $d$ , and  $N$  (independently of  $C$  and  $\ell$ ).

**$\mu$ -Semantic Security:** For any polynomial  $d$  and any two messages  $x_0, x_1$ , the following two distributions are  $\mu$ -indistinguishable:

$$\left\{ \begin{array}{l} \text{params} \leftarrow \text{MFHE.Setup}(1^\lambda, 1^{d(\lambda)}) \\ (\text{pk}, \text{sk}) \leftarrow \text{MFHE.KeyGen}(\text{params}) \end{array} : (\text{params}, \text{pk}, \text{MFHE.Enc}(\text{pk}, x_0)) \right\}_{\lambda \in \mathbb{N}},$$

$$\left\{ \begin{array}{l} \text{params} \leftarrow \text{MFHE.Setup}(1^\lambda, 1^{d(\lambda)}) \\ (\text{pk}, \text{sk}) \leftarrow \text{MFHE.KeyGen}(\text{params}) \end{array} : (\text{params}, \text{pk}, \text{MFHE.Enc}(\text{pk}, x_1)) \right\}_{\lambda \in \mathbb{N}}.$$

**Definition 2.26** (Threshold Multi-Key FHE). A threshold multi-key FHE scheme is a multi-key FHE scheme with the following two additional algorithms:

- **MFHE.PartDec** on input an expanded ciphertext  $\widehat{\text{ct}}$ , public keys  $\text{pk}_1, \dots, \text{pk}_N$ , an index  $i \in [N]$ , and the  $i$ th secret key  $\text{sk}_i$ , outputs a partial decryption  $p_i$ .

- $\text{MFHE.FinDec}$  on input partial decryptions  $p_1, \dots, p_N$ , outputs a message.

We further require correctness and simulatability of partial decryptions defined as follows: Let  $\text{params} \leftarrow \text{MFHE.Setup}(1^\lambda, 1^d)$ ,  $(\text{pk}_i, \text{sk}_i) \leftarrow \text{MFHE.KeyGen}(\text{params})$  for  $i \in [N]$ , and let  $x_1, \dots, x_\ell$  be messages. Further let  $I_1 \in [N], \dots, I_\ell \in [N]$ , and let  $\text{ct}_i \leftarrow \text{MFHE.Enc}(\text{pk}_{I_i}, x_i)$  and  $\widehat{\text{ct}}_i \leftarrow \text{MFHE.Expand}((\text{pk}_1, \dots, \text{pk}_N), I_i, \text{ct}_i)$  for  $i \in [\ell]$ . Finally let  $C$  be a circuit of depth at most  $d$  and let  $\widehat{\text{ct}} := \text{MFHE.Eval}(\text{params}, C, (\widehat{\text{ct}}_1, \dots, \widehat{\text{ct}}_\ell))$ . We then require:

**Correctness of Decryption:** For  $p_i \leftarrow \text{MFHE.PartDec}(\widehat{\text{ct}}, (\text{pk}_1, \dots, \text{pk}_N), i, \text{sk}_i)$ ,  $i \in [N]$ , we have  $\text{MFHE.FinDec}(p_1, \dots, p_N) = C(x_1, \dots, x_\ell)$  with probability 1.

**$\mu$ -Simulatability of Partial Decryptions:** There exists a PPT simulator  $\text{Sim}$  that on input  $i \in [N]$ ,  $(\text{sk}_j)_{j \in [N] \setminus \{i\}}$ ,  $\widehat{\text{ct}}$ , and  $C(x_1, \dots, x_\ell)$ , produces a simulated partial decryption  $p'_i$  such that

$$\delta(p_i, p'_i) \leq O(\mu(\lambda)),$$

where  $p_i \leftarrow \text{MFHE.PartDec}(\widehat{\text{ct}}, (\text{pk}_1, \dots, \text{pk}_N), i, \text{sk}_i)$ . Here, the randomness for the statistical distance is only over the coins of  $\text{Sim}$  and  $\text{MFHE.PartDec}$ , and all other values are fixed.

### 3 Pseudo Flawed-Smudging Generators

#### 3.1 Definitions

In this section, we 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 coefficients.

**Definition 3.1** (Smudging distributions). Let  $\ell$  be a positive integer, let  $\varepsilon \in [0, 1]$ , and let  $B$  either be a positive integer or an  $\ell$ -dimensional vector of positive integers. We say a distribution  $\mathcal{X}$  over  $\mathbb{Z}^\ell$  is  $(B, \varepsilon)$ -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 3.2** (Bit-fixing distributions). Let  $\mathcal{D}$  be a distribution over strings in  $\Delta^{\leq \ell}$  for some set  $\Delta$  and some integer  $\ell$ . 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  $\bar{x}$  from  $\mathcal{D}$  conditioned on  $\bar{x}_I = x$ . 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 3.3** (Flawed-smudging distributions). Let  $\ell$  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  $(K, \mu)$ -flawed-smudging

for  $\mathcal{E}$  if there exist randomized predicates  $\{\text{BAD}_i: \mathbb{Z}^{\ell+1} \rightarrow \{0, 1\}\}_{i \in [\ell]}$  such that the following two distributions are identical:

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

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

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

We say the distribution  $\mathcal{X}$  is  $(K, \mu)$ -flawed-smudging for  $B$ -bounded distributions if it is  $(K, \mu)$ -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 3.4.* We remark that in the above definition, we require  $\mathcal{D}_1$  and  $\mathcal{D}_2$  to be identically distributed for convenience. It can be relaxed to being  $\mu$ -statistically close. For our FE schemes, it suffices to have  $\mu$  being negligible, and for IO schemes,  $\mu$  needs to be subexponentially small.

*Remark 3.5.* A more direct generalization of the definition of smudging distributions (see [Definition 3.1](#)) 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 3.3](#) resolves this issue by sampling the non-bad coordinates freshly after sampling bad.

[Definition 3.3](#) asserts that  $(E, X + E, \text{bad})$  and  $(\bar{E}, X + E, \text{bad})$  are equally distributed. As we next show, this is equivalent to the non-bad coordinates of  $E$  being independent of  $X + E$  and bad (only depending on the bad coordinates of  $E$ ). This characterization will later be useful to prove properties of flawed-smudging distributions.

**Lemma 3.6.** *Let  $\ell$  be a positive integer and let  $\mathcal{X}$  and  $\mathcal{E}$  be distributions over  $\mathbb{Z}^\ell$ . Further let  $\text{BAD}_i: \mathbb{Z}^{\ell+1} \rightarrow \{0, 1\}$  be randomized predicates for  $i \in [\ell]$ , and let  $E \leftarrow \mathcal{E}$ ,  $X \leftarrow \mathcal{X}$ ,  $\text{bad} = (\text{bad}_i \leftarrow \text{BAD}_i(E_i, X))_{i \in [\ell]}$ , and  $\bar{E} \leftarrow \mathcal{E}|_{E_{\text{bad}}, \text{bad}}$ . Then, the distributions of  $(E, X + E, \text{bad})$  and  $(\bar{E}, X + E, \text{bad})$  are equal if and only if for all  $y, e \in \mathbb{Z}^\ell$  and for all  $I \subseteq [\ell]$ ,*

$$\Pr[X + E = y \wedge \text{bad} = I \wedge E = e] = \Pr[X + E = y \wedge \text{bad} = I \wedge E_I = e_I] \cdot \Pr[E_{[\ell] \setminus I} = e_{[\ell] \setminus I} \mid E_I = e_I].$$

*Proof.* We have for all  $y, e \in \mathbb{Z}^\ell$  and for all  $I \subseteq [\ell]$ ,

$$\begin{aligned} \Pr[X + E = y \wedge \text{bad} = I \wedge \bar{E} = e] &= \Pr[X + E = y \wedge \text{bad} = I \wedge \bar{E}_I = e_I] \\ &\quad \cdot \Pr[\bar{E}_{[\ell] \setminus I} = e_{[\ell] \setminus I} \mid X + E = y \wedge \text{bad} = I \wedge \bar{E}_I = e_I]. \end{aligned}$$

By definition of  $\bar{E}$ , we further have  $\bar{E}_I = E_I$  for  $\text{bad} = I$ , and  $\bar{E}_{[\ell] \setminus I}$  is sampled from the same distribution as  $E_{[\ell] \setminus I}$  conditioned on the values of  $E_I$  being fixed. Hence,

$$\Pr[\bar{E}_{[\ell] \setminus I} = e_{[\ell] \setminus I} \mid X + E = y \wedge \text{bad} = I \wedge \bar{E}_I = e_I] = \Pr[E_{[\ell] \setminus I} = e_{[\ell] \setminus I} \mid E_I = e_I].$$

This implies that

$$\begin{aligned} \Pr[X + E = y \wedge \text{bad} = I \wedge \bar{E} = e] \\ = \Pr[X + E = y \wedge \text{bad} = I \wedge E_I = e_I] \cdot \Pr[E_{[\ell] \setminus I} = e_{[\ell] \setminus I} \mid E_I = e_I]. \end{aligned}$$

Therefore,  $(E, X + E, \text{bad})$  and  $(\bar{E}, X + E, \text{bad})$  are equally distributed if and only if we have  $\Pr[X + E = y \wedge \text{bad} = I \wedge E = e] = \Pr[X + E = y \wedge \text{bad} = I \wedge \bar{E} = e]$  for all  $y, e$ , and  $I$ , which is the case if and only if the right hand side of the equation above is equal to  $\Pr[X + E = y \wedge \text{bad} = I \wedge E = e]$ .  $\square$

**Example.** We now give a very simple example of a flawed-smudging distribution. Let  $\ell$  be a positive integer, let  $\mathcal{X}$  be the uniform distribution over  $\{1, \dots, 10\}^\ell$ , and let  $\mathcal{E}$  be the uniform distribution over  $\{0, 1\}^\ell$ . We show that  $\mathcal{X}$  is flawed-smudging for  $\mathcal{E}$ . Let  $X \leftarrow \mathcal{X}$  and  $E \leftarrow \mathcal{E}$ . Intuitively, we have for each coordinate  $i$  that the sum  $X_i + E_i$  hides  $E_i$  if  $X_i + E_i \in \{2, \dots, 10\}$ , because such sums are possible for both values of  $E$ . We therefore define the predicate

$$\text{BAD}_i(E_i, X) = \begin{cases} 0, & X_i + E_i \in \{2, \dots, 10\}, \\ 1, & X_i + E_i \in \{1, 11\}. \end{cases}$$

Now let  $\text{bad} = (\text{bad}_i \leftarrow \text{BAD}_i(E_i, X))_{i \in [\ell]}$  and let  $\bar{E} \leftarrow \mathcal{E}|_{E_{\text{bad}}, \text{bad}}$ . We have to show that the distributions of  $(E, X + E, \text{bad})$  and  $(\bar{E}, X + E, \text{bad})$  are equal. Since all coordinates of  $X, E, \bar{E}$ , and  $\text{bad}$  are independent, we can analyze these distributions coordinate-wise. For all  $e_i$  and  $y_i \in \{1, 11\}$ , we have that  $\text{bad}_i = 1$  if  $X_i + E_i = y_i$ , and therefore  $\bar{E}_i = E_i$ . Hence,

$$\Pr[\bar{E}_i = e_i \wedge X_i + E_i = y_i \wedge \text{bad}_i = 1] = \Pr[E_i = e_i \wedge X_i + E_i = y_i \wedge \text{bad}_i = 1],$$

and both probabilities are 0 for  $\text{bad}_i = 0$ . For all  $e_i$  and all  $y_i \in \{2, \dots, 10\}$ , we have  $y_i - e_i \in \{1, \dots, 10\}$ , which implies that  $\Pr[X_i + e_i = y_i] = \frac{1}{10}$ . Hence,

$$\Pr[E_i = e_i \wedge X_i + E_i = y_i \wedge \text{bad}_i = 0] = \Pr[E_i = e_i] \cdot \Pr[X_i + e_i = y_i] = \frac{1}{2} \cdot \frac{1}{10} = \frac{1}{20}.$$

Furthermore,  $\bar{E}_i$  for  $\text{bad}_i = 0$  is distributed independently and identically to  $E_i$ , which yields

$$\begin{aligned} \Pr[\bar{E}_i = e_i \wedge X_i + E_i = y_i \wedge \text{bad}_i = 0] &= \Pr[E_i = e_i] \cdot \Pr[X_i + E_i = y_i] \\ &= \Pr[E_i = e_i] \cdot \sum_{e'_i \in \{0, 1\}} \Pr[E_i = e'_i] \cdot \Pr[X_i + e'_i = y_i] \\ &= 1/20. \end{aligned}$$

We can therefore conclude that  $(E, X + E, \text{bad})$  and  $(\bar{E}, X + E, \text{bad})$  are equally distributed.

For each  $i \in [\ell]$ ,

$$\begin{aligned} \Pr[\text{bad}_i = 1] &= \Pr[X_i + E_i = 1] + \Pr[X_i + E_i = 11] \\ &= \Pr[X_i = 1 \wedge E_i = 0] + \Pr[X_i = 10 \wedge E_i = 1] \\ &= 1/10. \end{aligned}$$

We can therefore use the Chernoff-Hoeffding bound to obtain that for all  $\varepsilon > 0$ ,

$$\Pr[|\text{bad}|_1 \geq \ell \cdot (1/10 + \varepsilon)] \leq \exp(-2\varepsilon^2 \ell).$$

Hence,  $\mathcal{X}$  is  $(K, \mu)$ -flawed-smudging for the distribution  $\mathcal{E}$ , with  $K = (1/10 + \varepsilon)\ell$  and  $\mu = 1 - \exp(-2\varepsilon^2\ell)$ .

The crucial properties of  $\mathcal{X}$  here are that all coordinates of  $\mathcal{X}$  are independent and that the marginal distributions of each coordinate are smudging for the coordinates of  $\mathcal{E}$ . In [Section 3.2.5](#), we generalize this example by showing that if  $\mathcal{X} = \mathcal{X}_1 \times \dots \times \mathcal{X}_\ell$  is a product distribution and each  $\mathcal{X}_i$  is  $(B, \varepsilon)$ -smudging, then  $\mathcal{X}$  is flawed-smudging for  $B$ -bounded distributions.

**Pseudo flawed-smudging generators.** We finally 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 3.7.** (Pseudo Flawed-Smudging Generator) Let  $n, m, K, B$  be polynomials. A family of  $(K, \mu)$ -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  $(\text{PFG}, \mathcal{D}^{sd})$  in the support of  $\mathcal{PFG}_\lambda$  defines a function  $\text{PFG}: \mathbb{Z}^{n(\lambda)} \rightarrow \mathbb{Z}^{m(\lambda)}$  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  $(\text{PFG}, \mathcal{D}^{sd}) \in \text{Support}(\mathcal{PFG}_\lambda)$  and  $sd \in \text{Support}(\mathcal{D}^{sd})$ ,  $M(\text{PFG}, sd)$  runs in time  $\text{poly}(\lambda)$  and we have  $M(\text{PFG}, sd) = \text{PFG}(sd)$ . Furthermore,  $\mathcal{PFG}$  and all  $\mathcal{D}^{sd}$  in the support of  $\mathcal{PFG}_\lambda$  are efficiently samplable.

**$(K, \mu)$ -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  $\mu$ -indistinguishable:

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

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

*Remark 3.8.* We have defined PFGs to compute functions  $\text{PFG}: \mathbb{Z}^n \rightarrow \mathbb{Z}^m$ . In [Section 5](#), we will need PFGs that can be computed by polynomials over  $\mathbb{Z}_p$  for some  $p \in \mathbb{N}$ . There are two ways how this can fit our definition: Either the PFGs consist of polynomials over  $\mathbb{Z}$  that have bounded seeds and outputs, such that computing them modulo  $p$  does not cause any wrap-around. Or the PFGs already consist of polynomials over  $\mathbb{Z}_p$ , which can be viewed as computing a function (which is not a polynomial) over  $\mathbb{Z}$ .

## 3.2 Properties of (Flawed-)Smudging Distributions

In this section, we prove some properties of smudging and of flawed-smudging distributions.

### 3.2.1 Bounds on the Smudging Property

We show that a polynomially bounded distribution cannot hide a value entirely. To this end, we first prove that the min-entropy gives rise to an upper bound on the smudging property of a one-dimensional distribution.

**Lemma 3.9.** *Let  $X$  be a random variable over  $\mathbb{Z}$  with finite support  $S$ . We then have*

$$\delta(X, X + 1) \geq \max_{x \in S} \Pr[X = x].$$

*Proof.* Let  $S^\uparrow := \{x \in \mathbb{Z} \mid \Pr[X = x] > \Pr[X + 1 = x]\}$  and  $S^\downarrow := \{x \in \mathbb{Z} \mid \Pr[X = x] < \Pr[X + 1 = x]\}$ . By definition of the statistical distance, we have

$$\begin{aligned} \delta(X, X + 1) &= \frac{1}{2} \sum_{x \in \mathbb{Z}} |\Pr[X = x] - \Pr[X + 1 = x]| \\ &\geq \frac{1}{2} \sum_{x \in S^\uparrow} |\Pr[X = x] - \Pr[X + 1 = x]| + \frac{1}{2} \sum_{x \in S^\downarrow} |\Pr[X = x] - \Pr[X + 1 = x]|. \end{aligned}$$

Since the support  $S$  is finite, there are  $x_0, x_1 \in S^\uparrow$  such that  $\Pr[X + 1 = x_0] = 0$  and  $\Pr[X = x_1] = \max_{x \in S} \Pr[X = x]$ . Hence,  $\sum_{x \in S^\uparrow} |\Pr[X = x] - \Pr[X + 1 = x]| \geq \max_{x \in S} \Pr[X = x]$ . Analogously,  $\sum_{x \in S^\downarrow} |\Pr[X = x] - \Pr[X + 1 = x]| \geq \max_{x \in S} \Pr[X = x]$ . In conclusion, we have  $\delta(X, X + 1) \geq \max_{x \in S} \Pr[X = x]$ .  $\square$

The next lemma implies that polynomially bounded distributions cannot be smudging with negligible  $\varepsilon$ . This means they cannot completely hide a value.

**Lemma 3.10.** *Let  $\ell$  and  $B$  be positive integers, and let  $\mathcal{X}$  be a  $B$ -bounded distribution over  $\mathbb{Z}^\ell$ . Further assume that  $\mathcal{X}$  is  $(B', \varepsilon)$ -smudging for some  $\varepsilon \in [0, 1]$  and  $B' \geq 1$ . Then,*

$$\varepsilon \geq \frac{1}{2B + 1}.$$

*Proof.* Let  $X = (X_1, \dots, X_\ell) \leftarrow \mathcal{X}$ . Since  $\mathcal{X}$  is  $(B', \varepsilon)$ -smudging, we have  $\delta(X, X + e) \leq \varepsilon$  for all  $e \in [-B', B']^\ell \cap \mathbb{Z}^\ell$ . In particular,  $\delta(X, X + e_1) \leq \varepsilon$  for  $e_1 := (1, 0, \dots, 0)^\top$ . We therefore have

$$\varepsilon \geq \delta(X, X + e_1) \geq \delta(X_1, X_1 + 1).$$

Since the support of  $X_1$  is contained in  $[-B, B]$ , there exists some  $x_1 \in \mathbb{Z}$  such that  $\Pr[X_1 = x_1] \geq 1/(2B + 1)$ . Hence, [Lemma 3.9](#) implies that  $\delta(X_1, X_1 + 1) \geq 1/(2B + 1)$ , which concludes the proof.  $\square$

### 3.2.2 Preservation Under Addition of Independent Values

We show that if the distribution of  $X$  is (flawed-)smudging and  $Y$  is independent from  $X$ , then the distribution of  $X + Y$  is (flawed-)smudging. In other words, adding an independent value cannot destroy the (flawed-)smudging property.

**Lemma 3.11.** *Let  $\ell$  be a positive integer and let  $X$  and  $Y$  be independent random variables over  $\mathbb{Z}^\ell$ . If the distribution of  $X$  is  $(B, \varepsilon)$ -smudging, then the distribution of  $X + Y$  is  $(B, \varepsilon)$ -smudging.*

*Proof.* Since adding an independent random variable cannot increase the statistical distance, we have for all  $B$ -bounded  $e \in \mathbb{Z}^\ell$ ,

$$\delta(X + Y, X + Y + e) \leq \delta(X, X + e) \leq \varepsilon.$$

Hence, the distribution of  $X + Y$  is  $(B, \varepsilon)$ -smudging.  $\square$

**Lemma 3.12.** *Let  $\ell$  be a positive integer and let  $\mathcal{X}$  and  $\mathcal{E}$  be distributions over  $\mathbb{Z}^\ell$ . Further let  $X$  be a random variable with distribution  $\mathcal{X}$  and let  $Y$  be an independent random variable over  $\mathbb{Z}^\ell$ . Assume that  $\mathcal{X}$  is  $(K, \mu)$ -flawed-smudging for  $\mathcal{E}$ . Then, the distribution of  $X + Y$  is  $(K, \mu)$ -flawed-smudging for  $\mathcal{E}$ .*

*Proof.* By assumption, there are randomized predicates  $\{\text{BAD}_i : \mathbb{Z}^{\ell+1} \rightarrow \{0, 1\}\}_{i \in [\ell]}$  such that  $\mathcal{D}_1$  and  $\mathcal{D}_2$  as defined in Definition 3.3 are identical and  $|\text{bad}|_1 \leq K$  with probability at least  $1 - \mu$ . Now define  $\text{BAD}'_i$  for the distribution of  $X + Y$  as follows: Given  $E_i$  and  $X + Y$ , sample  $X'$  and  $Y'$  with marginal distributions equal to the distributions of  $X$  and  $Y$ , respectively, conditioned on  $X' + Y' = X + Y$ , and return  $\text{BAD}'_i(E_i, X')$ . Let  $\text{bad}' = (\text{bad}'_i \leftarrow \text{BAD}'_i(E_i, X'))_{i \in [\ell]}$  and let  $\bar{E}' \leftarrow \mathcal{E}|_{E_{\text{bad}'}, \text{bad}'}$ . Since  $X$  and  $X'$  have the same distribution and  $\mathcal{D}_1$  and  $\mathcal{D}_2$  are equal, we also have that  $(E, E + X', \text{bad}')$  and  $(\bar{E}', E + X', \text{bad}')$  have the same distribution. Note that  $Y'$  is independent of  $(E, E + X', \text{bad}')$  and  $(\bar{E}', E + X', \text{bad}')$ , since it only depends on  $X'$  given  $X + Y$ , and nothing depends on  $X + Y$  beyond depending on  $X'$ . Hence,  $(E, E + X' + Y', \text{bad}')$  and  $(\bar{E}', E + X' + Y', \text{bad}')$  also have the same distribution. Using  $X' + Y' = X + Y$ , we obtain that  $\mathcal{D}'_1$  and  $\mathcal{D}'_2$  defined as in Definition 3.3 for  $X + Y$  and  $\text{BAD}'$  are equal. Furthermore,  $\text{bad}$  and  $\text{bad}'$  are equally distributed, which implies that  $|\text{bad}'|_1 \leq K$  with probability at least  $1 - \mu$ . Thus, the distribution of  $X + Y$  is  $(K, \mu)$ -flawed-smudging for  $\mathcal{E}$ .  $\square$

### 3.2.3 Mixtures of (Flawed-)Smudging Distributions

We next show that the probabilistic mixture of several (flawed-)smudging distributions is also (flawed-)smudging.

**Lemma 3.13.** *Let  $\ell$  and  $k$  be positive integers, let for  $i \in [k]$ ,  $\varepsilon_i \in [0, 1]$  and let  $X_i$  be a random variable over  $\mathbb{Z}^\ell$  with  $(B, \varepsilon_i)$ -smudging distribution. Further let  $\alpha_1, \dots, \alpha_k \in [0, 1]$  such that  $\sum_{i=1}^k \alpha_i = 1$ , and define the random variable  $X$  with  $\Pr[X = x] = \sum_{i=1}^k \alpha_i \Pr[X_i = x]$ . Then, the distribution of  $X$  is  $(B, \sum_{i=1}^k \alpha_i \varepsilon_i)$ -smudging.*

*Proof.* Let  $e \in \mathbb{Z}^\ell$  be  $B$ -bounded. We then have

$$\begin{aligned} \delta(X, X + e) &= \frac{1}{2} \sum_{x \in \mathbb{Z}^\ell} |\Pr[X = x] - \Pr[X + e = x]| \\ &\leq \frac{1}{2} \sum_{x \in \mathbb{Z}^\ell} \sum_{i=1}^k \alpha_i \cdot |\Pr[X_i = x] - \Pr[X_i + e = x]| \\ &= \sum_{i=1}^k \alpha_i \cdot \delta(X_i, X_i + e). \end{aligned}$$

Since  $\delta(X_i, X_i + e) \leq \varepsilon_i$ , the claim follows.  $\square$

**Lemma 3.14.** *Let  $\ell$  and  $k$  be positive integers, let  $\mathcal{E}$  be a distribution over  $\mathbb{Z}^\ell$ , let for  $i \in [k]$ ,  $\mu_i \in [0, 1]$ , and let  $X_i$  be a random variable over  $\mathbb{Z}^\ell$  with  $(K, \mu_i)$ -flawed-smudging distribution for  $\mathcal{E}$ . Further let  $\alpha_1, \dots, \alpha_k \in [0, 1]$  such that  $\sum_{i=1}^k \alpha_i = 1$ , and define the random variable  $X$  with  $\Pr[X = x] = \sum_{i=1}^k \alpha_i \Pr[X_i = x]$ . Then, the distribution of  $X$  is  $(K, \sum_{i=1}^k \alpha_i \mu_i)$ -flawed-smudging for  $\mathcal{E}$ .*

*Proof.* By assumption, there are randomized predicates  $\{\text{BAD}_j^{(i)} : \mathbb{Z}^{\ell+1} \rightarrow \{0, 1\}\}_{j \in [\ell]}$  for  $i \in [k]$  such that  $\mathcal{D}_1^{(i)}$  and  $\mathcal{D}_2^{(i)}$  as defined in Definition 3.3 for  $X_i$  are identical and  $|\text{bad}^{(i)}|_1 \leq K$  with probability at least  $1 - \mu_i$ . Now define  $\text{BAD}'_j$  for the distribution of  $X$  as follows: Given  $E_j$  and  $X$ , sample  $(X'_1, \dots, X'_k, A)$  conditioned on  $X'_A = X$ , where  $X'_i$  is distributed as  $X_i$  for  $i \in [k]$  and  $\Pr[A = i] = \alpha_i$ . Then output  $\text{BAD}'_j(A)(E_j, X'_A)$ . Let  $\text{bad}' = (\text{bad}'_j \leftarrow \text{BAD}'_j(E_j, X))_{j \in [\ell]}$  and let  $\bar{E}' \leftarrow \mathcal{E}|_{E_{\text{bad}', \text{bad}'}}$  and  $\bar{E}^{(i)} \leftarrow \mathcal{E}|_{E_{\text{bad}^{(i)}, \text{bad}^{(i)}}$ .

We have for all  $t$

$$\begin{aligned} \Pr[(\bar{E}', E + X, \text{bad}') = t] &= \sum_{i=1}^k \Pr[A = i] \cdot \Pr[(\bar{E}', E + X'_A, \text{bad}') = t \mid A = i] \\ &= \sum_{i=1}^k \Pr[A = i] \cdot \Pr[(\bar{E}^{(i)}, E + X_i, \text{bad}^{(i)}) = t] \\ &= \sum_{i=1}^k \Pr[A = i] \cdot \Pr[(E, E + X_i, \text{bad}^{(i)}) = t] \\ &= \Pr[(E, E + X, \text{bad}') = t]. \end{aligned}$$

This established the desired equality of distributions.

Using that the distribution of  $\text{bad}'$  conditioned on  $A = i$  equals the distribution of  $\text{bad}^{(i)}$ , we obtain

$$\Pr[|\text{bad}'|_1 \leq K] = \sum_{i=1}^k \Pr[A = i] \cdot \Pr[|\text{bad}^{(i)}|_1 \leq K] \geq \sum_{i=1}^k \alpha_i \cdot (1 - \mu_i) = 1 - \sum_{i=1}^k \alpha_i \cdot \mu_i. \quad \square$$

### 3.2.4 Composition of Flawed-Smudging Distributions

We now show that the product of flawed-smudging distributions is again flawed-smudging, not only for product distributions but also for distributions  $\mathcal{E}$  with correlated coordinates. We prove this only for the product of two distributions, but the result can easily be extended to arbitrarily many distributions by induction.

**Lemma 3.15.** *Let  $\ell^{(1)}$  and  $\ell^{(2)}$  be positive integers, let  $\ell := \ell^{(1)} + \ell^{(2)}$ , and let  $\mathcal{E}$  be a distribution over  $\mathbb{Z}^\ell$ . Let  $E \leftarrow \mathcal{E}$ , and denote by  $E^{(1)}$  the first  $\ell^{(1)}$  coordinates of  $E$ , and by  $E^{(2)}$  the last  $\ell^{(2)}$  coordinates of  $E$ . Moreover, let  $\mathcal{X}^{(1)}$  and  $\mathcal{X}^{(2)}$  be distributions over  $\mathbb{Z}^{\ell^{(1)}}$  and  $\mathbb{Z}^{\ell^{(2)}}$ , respectively, such that  $\mathcal{X}^{(i)}$  is  $(K^{(i)}, \mu^{(i)})$ -flawed-smudging for the distribution of  $E^{(i)}$ . Then,  $\mathcal{X} := \mathcal{X}^{(1)} \times \mathcal{X}^{(2)}$  is  $(K^{(1)} + K^{(2)}, \mu^{(1)} + \mu^{(2)})$ -flawed-smudging for  $\mathcal{E}$ .*

*Proof.* Let  $\text{BAD}_i^{(1)}$  and  $\text{BAD}_i^{(2)}$  be the randomized predicates guaranteed by the flawed-smudging properties of  $\mathcal{X}^{(1)}$  and  $\mathcal{X}^{(2)}$ , respectively. To prove the flawed-smudging property of  $\mathcal{X}$ , we define the randomized predicates  $\text{BAD}_i$  for  $i \in [\ell]$  as follows: for  $i \leq \ell^{(1)}$ , evaluate  $\text{BAD}_i^{(1)}$ , and for  $i > \ell^{(1)}$ , evaluate  $\text{BAD}_{i-\ell^{(1)}}^{(2)}$ .

Let  $(X^{(1)}, X^{(2)}) \leftarrow \mathcal{X}^{(1)} \times \mathcal{X}^{(2)}$ ,  $\text{bad}^{(j)} = (\text{bad}_i^{(j)} \leftarrow \text{BAD}_i^{(j)}(E_i^{(j)}, X^{(j)}))_{i \in [\ell^{(j)}]}$ ,  $\text{bad} = (\text{bad}_i \leftarrow \text{BAD}_i(E_i, X))_{i \in [\ell]}$ , and  $\bar{E} \leftarrow \mathcal{E}|_{E_{\text{bad}, \text{bad}}}$ . We have to show that  $(E, X + E, \text{bad})$  and  $(\bar{E}, X + E, \text{bad})$  have the same distribution. Let  $y, e \in \mathbb{Z}^\ell$ ,  $I \subseteq [\ell]$ , let  $y^{(1)}, y^{(2)}$ , and  $e^{(1)}, e^{(2)}$

be the first  $\ell^{(1)}$  and last  $\ell^{(2)}$  coordinates of  $y$  and  $e$ , respectively, and let  $I^{(1)} := I \cap [\ell^{(1)}]$  and  $I^{(2)} := I \cap \{\ell^{(1)} + 1, \dots, \ell\}$ . Because  $X^{(2)}$  and  $\text{bad}^{(2)}$  do not depend on  $X^{(1)}$  and  $\text{bad}^{(1)}$ , we have

$$\begin{aligned} & \Pr[E = e \wedge X + E = y \wedge \text{bad} = I] \\ &= \Pr[E = e \wedge X^{(1)} + E^{(1)} = y^{(1)} \wedge \text{bad}^{(1)} = I^{(1)}] \\ & \quad \cdot \Pr[X^{(2)} + E^{(2)} = y^{(2)} \wedge \text{bad}^{(2)} = I^{(2)} \mid E = e] \\ &= \Pr[E = e] \cdot \prod_{j=1}^2 \Pr[X^{(j)} + E^{(j)} = y^{(j)} \wedge \text{bad}^{(j)} = I^{(j)} \mid E = e]. \end{aligned}$$

Since  $X^{(j)}$  and  $\text{bad}^{(j)}$  do not depend on  $E^{(j')}$  for  $j' \neq j$ , and using [Lemma 3.6](#) together with the fact that the distribution of  $X^{(j)}$  is flawed-smudging for the distribution of  $E^{(j)}$ , we obtain

$$\begin{aligned} & \Pr[X^{(j)} + E^{(j)} = y^{(j)} \wedge \text{bad}^{(j)} = I^{(j)} \mid E = e] \\ &= \Pr[X^{(j)} + E^{(j)} = y^{(j)} \wedge \text{bad}^{(j)} = I^{(j)} \mid E^{(j)} = e^{(j)}] \\ &= \Pr[X^{(j)} + E^{(j)} = y^{(j)} \wedge \text{bad}^{(j)} = I^{(j)} \mid E_{I^{(j)}}^{(j)} = e_{I^{(j)}}^{(j)}]. \end{aligned}$$

Hence,

$$\begin{aligned} & \Pr[E_I = e_I \wedge X + E = y \wedge \text{bad} = I] \\ &= \sum_{\substack{e_i \in \mathbb{Z} \\ i \in [\ell] \setminus I}} \Pr[E = e \wedge X + E = y \wedge \text{bad} = I] \\ &= \sum_{\substack{e_i \in \mathbb{Z} \\ i \in [\ell] \setminus I}} \Pr[E = e] \cdot \prod_{j=1}^2 \Pr[X^{(j)} + E^{(j)} = y^{(j)} \wedge \text{bad}^{(j)} = I^{(j)} \mid E_{I^{(j)}}^{(j)} = e_{I^{(j)}}^{(j)}] \\ &= \Pr[E_I = e_I] \cdot \prod_{j=1}^2 \Pr[X^{(j)} + E^{(j)} = y^{(j)} \wedge \text{bad}^{(j)} = I^{(j)} \mid E_{I^{(j)}}^{(j)} = e_{I^{(j)}}^{(j)}]. \end{aligned}$$

This implies

$$\frac{\Pr[E = e \wedge X + E = y \wedge \text{bad} = I]}{\Pr[E_I = e_I \wedge X + E = y \wedge \text{bad} = I]} = \Pr[E_{[\ell] \setminus I} = e_{[\ell] \setminus I} \mid E_I = e_I].$$

Again using [Lemma 3.6](#) yields that  $(E, X + E, \text{bad})$  and  $(\bar{E}, X + E, \text{bad})$  are equally distributed.

Note that  $|\text{bad}|_1 = |\text{bad}^{(1)}|_1 + |\text{bad}^{(2)}|_1$ . Hence,

$$\begin{aligned} \Pr[|\text{bad}|_1 > K^{(1)} + K^{(2)}] &\leq \Pr[|\text{bad}^{(1)}|_1 > K^{(1)} \vee |\text{bad}^{(2)}|_1 > K^{(2)}] \\ &\leq \Pr[|\text{bad}^{(1)}|_1 > K^{(1)}] + \Pr[|\text{bad}^{(2)}|_1 > K^{(2)}] \\ &\leq \mu^{(1)} + \mu^{(2)}. \end{aligned}$$

This implies that  $\mathcal{X}$  is  $(K^{(1)} + K^{(2)}, \mu^{(1)} + \mu^{(2)})$ -flawed-smudging for  $\mathcal{E}$ -bounded and concludes the proof.  $\square$

### 3.2.5 Smudging and Independence Implies Flawed-Smudging

We want to show that if several mutually independent random variables each have a smudging distribution, then their joint distribution is flawed-smudging. To this end, we first prove a general lemma. Our starting point is the known fact that for two random variables  $X$  and  $X'$ , one can define events  $\text{bad}$  and  $\text{bad}'$  such that the probability of these events equals  $\delta(X, X')$  each and the distribution of  $X$  conditioned on  $\neg\text{bad}$  is equal to the distribution of  $X'$  conditioned on  $\neg\text{bad}'$  [MPR07]. We are interested in the following related statement, for which the  $\text{bad}$  event can be defined similarly: Assume  $X$  and  $E$  are independent random variables over  $\mathbb{Z}$  such that  $E$  is  $B$ -bounded and  $\delta(X, X + e)$  is “small” for all  $e$  in the support of  $E$ . Then, there exists an event  $\text{bad}$  with “small” probability such that if  $\text{bad}$  does not occur, then  $E$  does not depend on  $X + E$  and the fact that  $\text{bad}$  does not occur.

The high-level idea is to define the event  $\text{bad}$  such that for all  $y \in \mathbb{Z}$ ,  $\Pr[X + E = y \wedge \neg\text{bad}] = \min_e \{\Pr[X + e = y]\}$ . This means that given  $X + E = y$ , the probability of  $\text{bad}$  gets larger the greater the gap between the probabilities of  $X + e = y$  for different values of  $e$  is. The following lemma generalizes this to several  $X_i$  and  $E_i$ . We define events  $\text{bad}_i$  for each  $X_i + E_i$  and get the statement that intuitively says that the coordinates of  $E$  that are not  $\text{bad}$  do not depend on any  $X_i + E_i$ .

**Lemma 3.16.** *Let  $\ell \in \mathbb{N}$  and let  $X_1, \dots, X_\ell, E_1, \dots, E_\ell$  be random variables over  $\mathbb{Z}$  such that the  $\ell + 1$  random variables  $X_1, \dots, X_\ell$ , and  $(E_1, \dots, E_\ell)$  are mutually independent.<sup>6</sup> Further let  $R_i$  be the support of  $E_i$  with  $|R_i| < \infty$ . Then, there exist events  $\text{bad}_1, \dots, \text{bad}_\ell$  with  $\text{bad}_i = \text{BAD}_i(E_i, X_i)$  depending only on  $E_i$  and  $X_i$  such that for all  $y, e \in \mathbb{Z}^\ell$  and for all  $I \subseteq [\ell]$ ,*

- (i)  $\Pr[X + E = y \wedge \text{bad} = I \wedge E = e] = \Pr[X + E = y \wedge \text{bad} = I \wedge E_I = e_I] \cdot \Pr[E_{[\ell] \setminus I} = e_{[\ell] \setminus I} \mid E_I = e_I]$ ,
- (ii)  $\Pr[\bigwedge_{i \in I} \text{bad}_i] \leq \prod_{i \in I} 4 \cdot \sum_{e'_i \in R_i} \delta(X_i, X_i + e'_i)$ .

*Proof.* We define  $\text{BAD}_i$  for  $E_i = e_i$  and  $X_i = x_i$  as

$$\Pr[\text{BAD}_i(e_i, x_i) = 1] := 1 - \min_{e'_i \in R_i} \left\{ \frac{\Pr[X_i + e'_i = x_i + e_i]}{\Pr[X_i = x_i]} \right\}.$$

Note that this is a valid probability since  $\min_{e'_i \in R_i} \{\Pr[X_i + e'_i = x_i + e_i]\} \leq \Pr[X_i = x_i]$ . We then have for  $y, e \in \mathbb{Z}^\ell$  with  $\Pr[X + E = y \wedge E = e] > 0$ ,

$$\Pr[\text{bad}_i \mid X + E = y \wedge E = e] = 1 - \min_{e'_i \in R_i} \left\{ \frac{\Pr[X_i + e'_i = y_i]}{\Pr[X_i + e_i = y_i]} \right\},$$

where the  $\text{bad}_i$  are sampled independently given  $X + E = y \wedge E = e$ .

Now fix  $y, e \in \mathbb{Z}^\ell$  and  $I \subseteq [\ell]$ . We then have using the independence of the  $X_i$  and the

---

<sup>6</sup>But,  $E_1, \dots, E_\ell$  need not be independent.

conditional independence of the  $\text{bad}_i$ ,

$$\begin{aligned}
& \Pr[X + E = y \wedge E = e \wedge \text{bad} = I] \\
&= \Pr[E = e] \cdot \prod_{i \in [\ell]} \Pr[X_i + e_i = y_i] \cdot \prod_{i \in [\ell] \setminus I} \min_{e'_i \in R_i} \left\{ \frac{\Pr[X_i + e'_i = y_i]}{\Pr[X_i + e_i = y_i]} \right\} \\
&\quad \cdot \prod_{i \in I} \left( 1 - \min_{e'_i \in R_i} \left\{ \frac{\Pr[X_i + e'_i = y_i]}{\Pr[X_i + e_i = y_i]} \right\} \right) \tag{1} \\
&= \Pr[E = e] \cdot \prod_{i \in I} \Pr[X_i + e_i = y_i] \cdot \prod_{i \in [\ell] \setminus I} \min_{e'_i \in R_i} \{ \Pr[X_i + e'_i = y_i] \} \\
&\quad \cdot \prod_{i \in I} \left( 1 - \min_{e'_i \in R_i} \left\{ \frac{\Pr[X_i + e'_i = y_i]}{\Pr[X_i + e_i = y_i]} \right\} \right).
\end{aligned}$$

This implies

$$\begin{aligned}
& \Pr[X + E = y \wedge E_I = e_I \wedge \text{bad} = I] \\
&= \sum_{\substack{e_i \in \mathbb{Z} \\ i \in [\ell] \setminus I}} \Pr[X + E = y \wedge E = e \wedge \text{bad} = I] \\
&= \Pr[E_I = e_I] \cdot \prod_{i \in I} \Pr[X_i + e_i = y_i] \cdot \prod_{i \in [\ell] \setminus I} \min_{e'_i \in R_i} \{ \Pr[X_i + e'_i = y_i] \} \\
&\quad \cdot \prod_{i \in I} \left( 1 - \min_{e'_i \in R_i} \left\{ \frac{\Pr[X_i + e'_i = y_i]}{\Pr[X_i + e_i = y_i]} \right\} \right). \tag{2}
\end{aligned}$$

Dividing equation (1) by equation (2) yields

$$\Pr[E_{[\ell] \setminus I} = e_{[\ell] \setminus I} \mid X + E = y \wedge E_I = e_I \wedge \text{bad} = I] = \Pr[E_{[\ell] \setminus I} = e_{[\ell] \setminus I} \mid E_I = e_I].$$

Since  $\Pr[X + E = y \wedge \text{bad} = I \wedge E = e] = \Pr[X + E = y \wedge \text{bad} = I \wedge E_I = e_I] \cdot \Pr[E_{[\ell] \setminus I} = e_{[\ell] \setminus I} \mid X + E = y \wedge \text{bad} = I \wedge E_I = e_I]$ , this concludes the proof of the first claim of the lemma.

To prove (ii), note that  $\bigwedge_{i \in I} \text{bad}_i$  only depends on  $X_i$  and  $E_i$  for  $i \in I$ . Together with the independence of the  $X_i$ , this yields

$$\begin{aligned}
\Pr \left[ \bigwedge_{i \in I} \text{bad}_i \right] &= \sum_{\substack{y_i, e_i \in \mathbb{Z} \\ i \in I}} \Pr[X_I + E_I = y_I \wedge E_I = e_I] \cdot \Pr \left[ \bigwedge_{i \in I} \text{bad}_i \mid X_I + E_I = y_I \wedge E_I = e_I \right] \\
&= \sum_{\substack{y_i, e_i \in \mathbb{Z} \\ i \in I}} \Pr[X_I + E_I = y_I \wedge E_I = e_I] \cdot \prod_{i \in I} \left( 1 - \min_{e'_i \in R_i} \left\{ \frac{\Pr[X_i + e'_i = y_i]}{\Pr[X_i + e_i = y_i]} \right\} \right) \\
&= \sum_{\substack{y_i, e_i \in \mathbb{Z} \\ i \in I}} \Pr[E_I = e_I] \cdot \prod_{i \in I} \left( \Pr[X_i + e_i = y_i] - \min_{e'_i \in R_i} \{ \Pr[X_i + e'_i = y_i] \} \right) \\
&\leq \sum_{\substack{y_i, e_i \in \mathbb{Z} \\ i \in I}} \Pr[E_I = e_I] \cdot \prod_{i \in I} \left( \max_{e'_i \in R_i} \Pr[X_i + e'_i = y_i] - \min_{e'_i \in R_i} \{ \Pr[X_i + e'_i = y_i] \} \right).
\end{aligned}$$

Since the term in the product over  $i \in I$  does not depend on  $e_I$ , we obtain

$$\begin{aligned}
\Pr \left[ \bigwedge_{i \in I} \text{bad}_i \right] &\leq \sum_{\substack{y_i \in \mathbb{Z} \\ i \in I}} \prod_{\substack{i \in I \\ e'_i \in R_i}} \left( \max_{e'_i \in R_i} \Pr[X_i + e'_i = y_i] - \min_{e'_i \in R_i} \{ \Pr[X_i + e'_i = y_i] \} \right) \\
&= \prod_{i \in I} \sum_{y_i \in \mathbb{Z}} \left( \max_{e'_i \in R_i} \Pr[X_i + e'_i = y_i] - \min_{e'_i \in R_i} \{ \Pr[X_i + e'_i = y_i] \} \right) \\
&\leq \prod_{i \in I} \sum_{y_i \in \mathbb{Z}} \left( \left| \max_{e'_i \in R_i} \Pr[X_i + e'_i = y_i] - \Pr[X_i = y_i] \right| \right. \\
&\quad \left. + \left| \Pr[X_i = y_i] - \min_{e'_i \in R_i} \{ \Pr[X_i + e'_i = y_i] \} \right| \right).
\end{aligned}$$

Since  $\left| \max_{e'_i \in R_i} \Pr[X_i + e'_i = y_i] - \Pr[X_i = y_i] \right|$  and  $\left| \Pr[X_i = y_i] - \min_{e'_i \in R_i} \{ \Pr[X_i + e'_i = y_i] \} \right|$  are both upper bounded by  $\sum_{e'_i \in R_i} |\Pr[X_i = y_i] - \Pr[X_i + e'_i = y_i]|$ , we have

$$\begin{aligned}
\Pr \left[ \bigwedge_{i \in I} \text{bad}_i \right] &\leq \prod_{i \in I} 2 \cdot \sum_{e'_i \in R_i, y_i \in \mathbb{Z}} |\Pr[X_i = y_i] - \Pr[X_i + e'_i = y_i]| \\
&= \prod_{i \in I} 4 \cdot \sum_{e'_i \in R_i} \delta(X_i, X_i + e'_i). \quad \square
\end{aligned}$$

Using the lemma above, we can now show that independent random variables with smudging distributions jointly have a flawed-smudging distribution.

**Proposition 3.17.** *Let  $\ell$  and  $K$  be positive integers and let  $\mathcal{X}$  be a distribution over  $\mathbb{Z}^\ell$  such that for  $(X_1, \dots, X_\ell) \leftarrow \mathcal{X}$ ,  $X_1, \dots, X_\ell$  are mutually independent and the distribution of each  $X_i$  is  $(B, \varepsilon)$ -smudging for  $\varepsilon \leq \frac{K+1}{22^\ell \cdot (2B+1)}$ . Further let  $\mu = 2^{-K}$ . Then,  $\mathcal{X}$  is  $(K, \mu)$ -flawed-smudging for  $B$ -bounded distributions.*

*Proof.* Let  $\mathcal{E}$  be a  $B$ -bounded distribution over  $\mathbb{Z}^\ell$  and let  $E \leftarrow \mathcal{E}$ . Further let  $\text{bad}_i$  be the events guaranteed by Lemma 3.16. Using Lemma 3.16 (i) and Lemma 3.6, we obtain that the distributions  $\mathcal{D}_1$  and  $\mathcal{D}_2$  from Definition 3.3 are equal.

By Lemma 3.16 (ii), we have for all  $I \subseteq [\ell]$ ,  $\Pr \left[ \bigwedge_{i \in I} \text{bad}_i \right] \leq \prod_{i \in I} 4 \cdot \sum_{e'_i \in R_i} \delta(X_i, X_i + e'_i)$ . By our assumptions on  $\mathcal{X}$  and  $E$ , we have  $\delta(X_i, X_i + e'_i) \leq \varepsilon$  for all  $e'_i \in R_i$  and  $|R_i| \leq 2B + 1$ . Thus,  $\Pr \left[ \bigwedge_{i \in I} \text{bad}_i \right] \leq (4(2B + 1)\varepsilon)^{|I|}$ . We therefore have

$$\begin{aligned}
\Pr[|\text{bad}|_1 > K] &= \sum_{k=K+1}^{\ell} \Pr[|\text{bad}|_1 = k] \\
&\leq \sum_{k=K+1}^{\ell} \sum_{\substack{I \subseteq [\ell] \\ |I|=k}} \Pr \left[ \bigwedge_{i \in I} \text{bad}_i \right] \leq \sum_{k=K+1}^{\ell} \sum_{\substack{I \subseteq [\ell] \\ |I|=k}} (4(2B + 1)\varepsilon)^k.
\end{aligned}$$

There are  $\binom{\ell}{k}$  subsets  $I \subseteq [\ell]$  of size  $k$ . Using the bound  $\binom{\ell}{k} \leq \left(\frac{\ell e}{k}\right)^k$ , where  $e$  is Euler's constant, we obtain

$$\Pr[|\text{bad}|_1 > K] \leq \sum_{k=K+1}^{\ell} \left( \frac{\ell e \cdot 4(2B + 1)\varepsilon}{k} \right)^k < \sum_{k=K+1}^{\ell} \left( \frac{11\ell \cdot (2B + 1)\varepsilon}{K + 1} \right)^k.$$

Let  $\alpha := \frac{11\ell \cdot (2B+1)\varepsilon}{K+1}$ . Since  $\varepsilon \leq \frac{K+1}{22\ell \cdot (2B+1)}$  by assumption, we have  $\alpha \leq 1/2$ . Using the formula  $\sum_{k=0}^n \alpha^k = \frac{1-\alpha^{n+1}}{1-\alpha}$  for the first  $n$  terms of the geometric series, we then have

$$\Pr[|\text{bad}|_1 > K] < \sum_{k=0}^{\ell} \alpha^k - \sum_{k=0}^K \alpha^k = \frac{\alpha^{K+1} - \alpha^{\ell+1}}{1-\alpha} \leq \left(\frac{1}{2}\right)^K.$$

Hence, the probability that  $|\text{bad}|_1 \leq K$  is at least  $1 - 2^{-K} = 1 - \mu$ . Altogether, we conclude that  $\mathcal{X}$  is  $(K, \mu)$ -flawed-smudging for  $B$ -bounded distributions.  $\square$

### 3.2.6 Hiding Function Inputs

If  $X$  has a flawed-smudging distribution for the distribution of  $E$ , then  $E + X$  hides  $E$  at all coordinates that are not bad. In our application in [Section 6.3](#),  $E$  is a function of a random input  $V$  and we want to hide coordinates of the input  $V$ . Furthermore, the function that computes  $E$  from  $V$  has a certain local structure in the sense that each coordinate of  $E$  depends only on a few coordinates of  $V$ , say  $E_i$  is a function of  $V_{\Phi_i}$  for some set of coordinates  $\Phi_i$ . Now, if  $X + E$  hides all coordinates of  $E$  not in some bad set  $I$ , this should intuitively also hide  $V$  at all coordinates on which no bad coordinate of  $E$  depend, i.e.,  $V$  at coordinates not in  $\Phi_I := \bigcup_{i \in I} \Phi_i$  should be hidden. The following lemma shows that this intuition holds.

**Lemma 3.18.** *Let  $\ell$  and  $m$  be positive integers and let  $\mathcal{V}$  be a distribution over  $\mathbb{Z}^m$ . Further let for  $i \in [\ell]$ ,  $\Phi_i \subseteq [m]$  and let  $E_i: \mathbb{Z}^{|\Phi_i|} \rightarrow \mathbb{Z}$  be functions. For  $I \subseteq [\ell]$ , let  $\Phi_I := \bigcup_{i \in I} \Phi_i$ . Define  $E: \mathbb{Z}^m \rightarrow \mathbb{Z}^\ell$  as*

$$E(v_1, \dots, v_m) = (E_1((v_j)_{j \in \Phi_1}), \dots, E_\ell((v_j)_{j \in \Phi_\ell})).$$

*Let  $\mathcal{X}$  be a distribution over  $\mathbb{Z}^\ell$  that is flawed-smudging for  $E(\mathcal{V})$ , and let  $\text{BAD}_i: \mathbb{Z}^{\ell+1} \rightarrow \{0, 1\}$  for  $i \in [\ell]$  be the randomized predicates guaranteed to exist by [Definition 3.3](#). Then, the following two distributions are identical:*

$$\mathcal{D}_1 = \left\{ \begin{array}{l} V \leftarrow \mathcal{V} \\ X \leftarrow \mathcal{X} \\ \text{bad} = (\text{bad}_i \leftarrow \text{BAD}_i(E_i(V_{\Phi_i}), X))_{i \in [\ell]} \end{array} : (V, E(V) + X, \text{bad}) \right\},$$

$$\mathcal{D}_2 = \left\{ \begin{array}{l} V \leftarrow \mathcal{V} \\ X \leftarrow \mathcal{X} \\ \text{bad} = (\text{bad}_i \leftarrow \text{BAD}_i(E_i(V_{\Phi_i}), X))_{i \in [\ell]} \\ \bar{V} \leftarrow \mathcal{V}|_{V_{\Phi_{\text{bad}}}, \Phi_{\text{bad}}} \end{array} : (\bar{V}, E(V) + X, \text{bad}) \right\}.$$

*Proof.* Let  $V \leftarrow \mathcal{V}$  and  $X \leftarrow \mathcal{X}$ . Slightly abusing notation, we denote by  $E$  also the random variable  $E = E(V)$ . Further let  $\text{bad} = (\text{bad}_i \leftarrow \text{BAD}_i(E_i, X))_{i \in [\ell]}$ , and let  $\bar{V} \leftarrow \mathcal{V}|_{V_{\Phi_{\text{bad}}}, \Phi_{\text{bad}}}$ . We have to show that  $(V, E(V) + X, \text{bad})$  and  $(\bar{V}, E(V) + X, \text{bad})$  have the same distribution. To prove this, we need the following two facts, which we first prove. The first fact states that non-bad coordinates of  $E$  do not depend on bad. The second claim states that coordinates of  $V$  on which no bad coordinates depend, are also independent of bad.

**Claim 3.19.** *For all  $e \in \mathbb{Z}^\ell$  and  $I \subseteq [\ell]$ ,*

$$\Pr[E = e \wedge \text{bad} = I] = \Pr[E_I = e_I \wedge \text{bad} = I] \cdot \Pr[E_{[\ell] \setminus I} = e_{[\ell] \setminus I} \mid E_I = e_I].$$

*Proof.* Let  $e \in \mathbb{Z}^\ell$  and  $I \subseteq [\ell]$ . [Lemma 3.6](#) implies that

$$\Pr[X+E = y \wedge \text{bad} = I \wedge E = e] = \Pr[X+E = y \wedge \text{bad} = I \wedge E_I = e_I] \cdot \Pr[E_{[\ell] \setminus I} = e_{[\ell] \setminus I} \mid E_I = e_I].$$

Summing both sides of the equation over all  $y \in \mathbb{Z}^\ell$  implies the claim.  $\square$

We next prove the second fact, which follows from the first one.

**Claim 3.20.** *For all  $v \in \mathbb{Z}^m$  and  $I \subseteq [\ell]$ ,*

$$\Pr[V = v \wedge \text{bad} = I] = \Pr[V_{\Phi_I} = v_{\Phi_I} \wedge \text{bad} = I] \cdot \Pr[V_{[m] \setminus \Phi_I} = v_{[m] \setminus \Phi_I} \mid V_{\Phi_I} = v_{\Phi_I}].$$

*Proof.* Let  $v \in \mathbb{Z}^m$ ,  $I \subseteq [\ell]$ , and let  $e := E(v)$ . Then,

$$\begin{aligned} \Pr[V = v \wedge \text{bad} = I] &= \Pr[V = v \wedge E = e \wedge \text{bad} = I] \\ &= \Pr[E = e \wedge \text{bad} = I] \cdot \Pr[V = v \mid E = e \wedge \text{bad} = I]. \end{aligned}$$

We further have  $\Pr[V = v \mid E = e \wedge \text{bad} = I] = \Pr[V = v \mid E = e]$  since  $V$  depends on  $\text{bad} = \text{bad}(X, E(V))$  only via  $E = E(V)$ . Using [Claim 3.19](#), we thus obtain

$$\begin{aligned} \Pr[V = v \wedge \text{bad} = I] &= \Pr[E_I = e_I \wedge \text{bad} = I] \cdot \Pr[E_{[\ell] \setminus I} = e_{[\ell] \setminus I} \mid E_I = e_I] \cdot \Pr[V = v \mid E = e] \\ &= \Pr[E_I = e_I \wedge \text{bad} = I] \cdot \frac{\Pr[E = e]}{\Pr[E_I = e_I]} \cdot \frac{\Pr[V = v \wedge E = e]}{\Pr[E = e]}. \end{aligned}$$

Since  $E = E(V)$  is uniquely determined by  $V$ , we have  $\Pr[V = v \wedge E = e] = \Pr[V = v]$ . Hence,

$$\Pr[V = v \wedge \text{bad} = I] = \Pr[\text{bad} = I \mid E_I = e_I] \cdot \Pr[V = v].$$

This implies

$$\begin{aligned} \Pr[V_{\Phi_I} = v_{\Phi_I} \wedge \text{bad} = I] &= \sum_{\substack{v_i \in \mathbb{Z} \\ i \in [m] \setminus \Phi_I}} \Pr[V = v \wedge \text{bad} = I] \\ &= \sum_{\substack{v_i \in \mathbb{Z} \\ i \in [m] \setminus \Phi_I}} \Pr[\text{bad} = I \mid E_I(V_{\Phi_I}) = e_I] \cdot \Pr[V = v] \\ &= \Pr[\text{bad} = I \mid E_I(V_{\Phi_I}) = e_I] \cdot \Pr[V_{\Phi_I} = v_{\Phi_I}]. \end{aligned}$$

We therefore obtain

$$\frac{\Pr[V = v \wedge \text{bad} = I]}{\Pr[V_{\Phi_I} = v_{\Phi_I} \wedge \text{bad} = I]} = \Pr[V_{[m] \setminus \Phi_I} = v_{[m] \setminus \Phi_I} \mid V_{\Phi_I} = v_{\Phi_I}],$$

which implies the claim.  $\square$

We now prove [Lemma 3.18](#). To this end, let  $y \in \mathbb{Z}^\ell$ ,  $v \in \mathbb{Z}^m$ , and let  $I \subseteq [\ell]$ . We then have

$$\Pr[X + E = y \wedge \text{bad} = I \wedge \bar{V} = v] = \sum_{e \in \mathbb{Z}^\ell} \Pr[X + E = y \wedge \text{bad} = I \wedge \bar{V} = v \wedge E = e].$$

Since  $E_I = E_I(V) = E_I(\bar{V})$  for  $\text{bad} = I$ , we can fix  $e_I := E_I(v)$  and sum only over the remaining indices:

$$\begin{aligned} & \Pr[X + E = y \wedge \text{bad} = I \wedge \bar{V} = v] \\ &= \sum_{\substack{e_i \in \mathbb{Z} \\ i \in [\ell] \setminus I}} \underbrace{\Pr[X + E = y \wedge \text{bad} = I \wedge E = e]}_{\text{Lemma 3.6} \Pr[X+E=y \wedge \text{bad}=I \wedge E_I=e_I] \cdot \Pr[E_{[\ell] \setminus I} = e_{[\ell] \setminus I} | E_I=e_I]} \cdot \Pr[\bar{V} = v \mid E = e \wedge \text{bad} = I]. \end{aligned}$$

Moreover, using [Claim 3.19](#), we obtain

$$\begin{aligned} \Pr[\bar{V} = v \mid E = e \wedge \text{bad} = I] &= \frac{\Pr[\bar{V} = v \wedge E = e \wedge \text{bad} = I]}{\Pr[E = e \wedge \text{bad} = I]} \\ &= \frac{\Pr[\bar{V} = v \wedge E = e \wedge \text{bad} = I]}{\Pr[E_I = e_I \wedge \text{bad} = I] \cdot \Pr[E_{[\ell] \setminus I} = e_{[\ell] \setminus I} \mid E_I = e_I]}. \end{aligned}$$

Hence,

$$\begin{aligned} & \Pr[X + E = y \wedge \text{bad} = I \wedge \bar{V} = v] \\ &= \sum_{\substack{e_i \in \mathbb{Z} \\ i \in [\ell] \setminus I}} \Pr[X + E = y \wedge \text{bad} = I \wedge E_I = e_I] \cdot \frac{\Pr[\bar{V} = v \wedge E = e \wedge \text{bad} = I]}{\Pr[E_I = e_I \wedge \text{bad} = I]} \\ &= \Pr[X + E = y \wedge \text{bad} = I \wedge E_I = e_I] \cdot \frac{\Pr[\bar{V} = v \wedge E_I = e_I \wedge \text{bad} = I]}{\Pr[E_I = e_I \wedge \text{bad} = I]}. \end{aligned}$$

Using that  $E_I = E_I(V) = E_I(\bar{V})$  is uniquely determined by  $\bar{V}$ , and  $e_I = E_I(v)$ , we obtain  $\Pr[\bar{V} = v \wedge E_I = e_I \wedge \text{bad} = I] = \Pr[\bar{V} = v \wedge \text{bad} = I]$ . By definition of  $\bar{V}$ , we have for  $\text{bad} = I$  that  $\bar{V}_{\Phi_I} = V_{\Phi_I}$  and the coordinates not in  $\Phi_I$  have the same distribution as these coordinates in  $V$  conditioned on  $V_{\Phi_I} = v_{\Phi_I}$ . Therefore,

$$\begin{aligned} & \Pr[\bar{V} = v \wedge \text{bad} = I] \\ &= \Pr[\bar{V}_{\Phi_I} = v_{\Phi_I} \wedge \text{bad} = I] \cdot \Pr[\bar{V}_{[m] \setminus \Phi_I} = v_{[m] \setminus \Phi_I} \mid \bar{V}_{\Phi_I} = v_{\Phi_I} \wedge \text{bad} = I] \\ &= \Pr[V_{\Phi_I} = v_{\Phi_I} \wedge \text{bad} = I] \cdot \Pr[V_{[m] \setminus \Phi_I} = v_{[m] \setminus \Phi_I} \mid V_{\Phi_I} = v_{\Phi_I}]. \end{aligned}$$

[Claim 3.20](#) implies that the last line of the equation above is equal to  $\Pr[V = v \wedge \text{bad} = I]$ . Together with the equation above, we obtain

$$\Pr[X + E = y \wedge \text{bad} = I \wedge \bar{V} = v] = \Pr[X + E = y \wedge \text{bad} = I \wedge E_I = e_I] \cdot \frac{\Pr[V = v \wedge \text{bad} = I]}{\Pr[E_I = e_I \wedge \text{bad} = I]}.$$

Now let  $e_{[\ell] \setminus I} := E_{[\ell] \setminus I}(V)$ . Then,  $\Pr[V = v \wedge \text{bad} = I] = \Pr[V = v \wedge E = e \wedge \text{bad} = I]$ . Using [Claim 3.19](#), we thus have

$$\frac{\Pr[V = v \wedge \text{bad} = I]}{\Pr[E_I = e_I \wedge \text{bad} = I]} = \frac{\Pr[V = v \wedge E = e \wedge \text{bad} = I]}{\Pr[E = e \wedge \text{bad} = I]} \cdot \Pr[E_{[\ell] \setminus I} = e_{[\ell] \setminus I} \mid E_I = e_I].$$

Moreover, [Lemma 3.6](#) implies that

$$\Pr[X + E = y \wedge \text{bad} = I \wedge E_I = e_I] \cdot \Pr[E_{[\ell] \setminus I} = e_{[\ell] \setminus I} \mid E_I = e_I] = \Pr[X + E = y \wedge \text{bad} = I \wedge E = e],$$

which yields

$$\Pr[X + E = y \wedge \text{bad} = I \wedge \bar{V} = v] = \Pr[X + E = y \wedge \text{bad} = I \wedge E = e] \cdot \Pr[V = v \mid E = e \wedge \text{bad} = I].$$

Since  $V$  does not depend on  $X$ , we have

$$\Pr[V = v \mid E = e \wedge \text{bad} = I] = \Pr[V = v \mid E = e \wedge \text{bad} = I \wedge X + E = y],$$

leading to

$$\Pr[X + E = y \wedge \text{bad} = I \wedge \bar{V} = v] = \Pr[X + E = y \wedge \text{bad} = I \wedge E = e \wedge V = v].$$

Finally, the right hand side of the equation above equals  $\Pr[X + E = y \wedge \text{bad} = I \wedge V = v]$  because  $E$  is uniquely determined by  $V$ , which concludes the proof.  $\square$

## 4 Partially-Hiding Functional Encryption

In this section, we define *Partially-Hiding Functional Encryption* (PHFE) and construct it for some special classes of functions. PHFE computes functions  $g(x, y)$  that take a public input  $x$  and a private input  $y$ , and guarantees that a collection of ciphertexts and secret keys of PHFE reveals only the outputs of the function and *all public inputs*, while hiding the private inputs. The concept of PHFE was introduced in [AJKS18] as *three-restricted FE* for the special case of cubic polynomials with three inputs. PHFE is a direct extension of the notion of Partially-Hiding Predicate Encryption (PHPE) of [GVW15] that interpolates attribute based encryption and predicate encryption. PHFE instead interpolates attribute based encryption and functional encryption — if the public input  $c$  is empty, PHFE is equivalent to functional encryption, and if  $g$  is such that it outputs  $y$  when some predicate  $P$  on  $c$  outputs 1, then PHFE is attribute based encryption. Like in [GVW15], we will consider functions  $g$  that perform only simple computation on the private input, but complex computation on the public input. In our case, the simple computation is quadratic and we construct PHFE from bilinear maps.

**Definition 4.1** (PHFE). A secret-key *partially-hiding* functional encryption scheme PHFE for a class  $\{\mathcal{F}_\lambda\}_{\lambda \in \mathbb{N}}$  of functions with domains  $\mathcal{X}_\lambda \times \mathcal{Y}_\lambda$  and ranges  $\mathcal{Z}_\lambda$  consists of the four PPT algorithms PHFE.Setup, PHFE.KeyGen, PHFE.Enc, and PHFE.Dec, satisfying the same syntax and correctness as functional encryption in Definition 2.16, and the following security:

**PHFE simulation security:** PHFE is *1-key  $\mu$ -Sel-PH-Sim-secure* if there is a PPT universal simulator PSim such that for every  $\{f_\lambda\}_{\lambda \in \mathbb{N}}$ ,  $\{x_\lambda, y_\lambda\}_{\lambda \in \mathbb{N}}$ , every polynomial  $t$ , and all  $\{x_{i,\lambda}, y_{i,\lambda}\}_{\lambda \in \mathbb{N}, i \in [t(\lambda)]}$  with  $f_\lambda \in \mathcal{F}_\lambda$ ,  $x_\lambda, x_{i,\lambda} \in \mathcal{X}_\lambda$  and  $y_\lambda, y_{i,\lambda} \in \mathcal{Y}_\lambda$ , the following distributions are  $\mu$ -indistinguishable:

$$\left\{ \begin{array}{l} \text{msk} \leftarrow \text{PHFE.Setup}(1^\lambda) \\ \text{sk} \leftarrow \text{PHFE.KeyGen}(\text{msk}, f_\lambda) \\ \text{ct} \leftarrow \text{PHFE.Enc}(\text{msk}, x_\lambda, y_\lambda) \\ \{\text{ct}_i \leftarrow \text{PHFE.Enc}(\text{msk}, x_{i,\lambda}, y_{i,\lambda})\}_{i \in [t(\lambda)]} \end{array} : \text{sk}, \text{ct}, \{\text{ct}_i\}_{i \in [t(\lambda)]} \right\}_{\lambda \in \mathbb{N}},$$

$$\left\{ \text{PSim} \left( f_\lambda, f_\lambda(x_\lambda, y_\lambda), x_\lambda, \{x_{i,\lambda}, y_{i,\lambda}\}_{i \in [t(\lambda)]} \right) \right\}_{\lambda \in \mathbb{N}}.$$

Note that the only weakening from full simulation security of FE is that  $x_\lambda$  is provided to the simulator  $\text{PSim}$ .

In this section, all FE schemes used and constructed rely on bilinear map groups where the SXDH assumption holds, reflected in that their setup algorithm receives the description  $\text{pp} = (p, G_1, G_2, G_T, \text{pair})$  of a bilinear map as input. These FE schemes satisfy only *correctness in the exponent* in the sense that the function outputs are computed in the exponent of the groups, and are only extractable if they reside in a polynomially-sized range. Correspondingly, the PHFE schemes constructed satisfy a stronger variant of the above security notion, where the simulator  $\text{PSim}$  only receives an encoding of the function output  $[f_\lambda(x_\lambda, y_\lambda)]_1$  in the first source group  $G_1$  of the bilinear map.<sup>7</sup> In addition, they satisfy linear efficiency in the sense that encryption time is linear in the input length. We now give a formal definition.

**Definition 4.2.** Let FE be a secret-key or public-key functional encryption scheme for  $\{\mathcal{F}_\lambda\}_{\lambda \in \mathbb{N}}$  with domain  $\mathbb{Z}_{p(\lambda)}^{N(\lambda)}$  and range  $\mathbb{Z}_{p(\lambda)}^{M(\lambda)}$ .

- FE is *correct in the exponent* if for all  $\lambda$ ,  $f \in \mathcal{F}_\lambda$ ,  $x \in \mathbb{Z}_p^N$ , and all  $\text{pp} = (p, G_1, G_2, G_T, \text{pair})$  describing a bilinear map,

$$\Pr \left[ \begin{array}{l} \text{msk} \leftarrow \text{FE.Setup}(1^\lambda, \text{pp}) \\ \text{ct} \leftarrow \text{FE.Enc}(\text{msk}, x) \\ \text{sk} \leftarrow \text{FE.KeyGen}(\text{msk}, f) \end{array} : [f(x)]_T = \text{FE.Dec}(\text{sk}, \text{ct}) \right] = 1.$$

- It has *linear efficiency* if the encryption time is  $\text{poly}(\lambda)N$ .

**Definition 4.3.** Let PHFE be a partially-hiding functional encryption scheme for  $\{\mathcal{F}_\lambda\}_{\lambda \in \mathbb{N}}$ . We say that PHFE is *strongly  $\mu$ -Sel-PH-Sim-secure* if it satisfies the same security requirements as in [Definition 4.1](#), except that the setup algorithm  $\text{PHFE.Setup}$  receives additionally  $\text{pp} = (p, G_1, G_2, G_T, \text{pair})$  describing a bilinear map and the simulator  $\text{PSim}$  receives

$$\left\{ \text{PSim} \left( f_\lambda, [f_\lambda(x_\lambda, y_\lambda)]_1, x_\lambda, \{x_{i,\lambda}, y_{i,\lambda}\}_{i \in [t(\lambda)]} \right) \right\}_{\lambda \in \mathbb{N}}.$$

#### 4.1 PHFE for Polynomials of Degree 3

Let  $p$  be an arbitrary modulus, and  $N, S$  polynomials. We start with constructing a PHFE scheme for the class  $\{\mathcal{G}_\lambda^{N,S}\}$  of degree 3 polynomials  $g(\mathbf{x}, \mathbf{y}, \mathbf{z})$  over  $\mathbb{Z}_p$  that are multilinear in each input vector  $\mathbf{x}, \mathbf{y}, \mathbf{z} \in \mathbb{Z}_p^N$  and have size  $S$ . The first input  $\mathbf{x}$  is the public input while the second and third  $\mathbf{y}, \mathbf{z}$  are private inputs.

Note that the computation over the private inputs is quadratic. In the literature, there are constructions of FE for quadratic polynomials from bilinear maps [[Lin17](#), [BCFG17](#)]. It turns out that a simple modification of the scheme by [[Lin17](#)] allows for adding the public input and performing a linear computation on it. The proof in [[Lin17](#)] establishes the indistinguishability security of their quadratic FE scheme, but a careful examination reveals that the scheme also satisfies simulation security as in [Definition 2.18](#). The same proof, slightly modified, establishes the simulation security of our degree 3 PHFE. For completeness, below we describe formally our construction and proof, while highlighting the difference from the construction of [[Lin17](#)].

<sup>7</sup>We can also swap  $G_1$  with  $G_2$  since they are symmetric.

**Ingredients.** Like in [Lin17], our construction uses a secret key Function Hiding IPE scheme constructed from bilinear map with the following canonical form, as the one in [Lin17].

**Theorem 4.4** ([Lin17]). *Assume the  $\mu$ -indistinguishability of the SXDH assumption over bilinear map groups of order  $p = p(\lambda)$ . There is  $\text{poly}(\mu)$ -Sel-FH IPE  $\{\mathbf{hIPE}^N\}$  for inner products over  $\mathbb{Z}_p$  with the following properties:*

**Canonical Form** *For every polynomial  $N$  and security parameter  $\lambda$ ,  $\mathbf{hIPE}^N$  using bilinear map groups  $(p, G_1, G_2, G_T, \text{pair})$  satisfy that for every  $\mathbf{x} \in \mathbb{Z}_p^N$ , a ciphertext (or secret key) of a vector  $\mathbf{x}$  consists of only encodings in the group  $G_1$  (or  $G_2$  respectively) of elements that depend linearly in the encoded vector  $\mathbf{x}$ , that is, of the form  $[L_1(\mathbf{x})]_1$  for some linear function  $L_1$  (or  $[L_2(\mathbf{x})]_2$ ).*

**Linear Efficiency** *For every polynomial  $N$  and security parameter  $\lambda$ , the encryption and key generation algorithms of  $\mathbf{hIPE}^N$  takes time  $\text{poly}(\lambda)N$ .*

**PHFE for Degree-3 Multilinear Polynomial.** Let  $\mathbf{hIPE} = (\text{hIPE.Setup}, \text{hIPE.KeyGen}, \text{hIPE.Enc}, \text{hIPE.Dec})$ ; we omit the input lengths since they are implicit in the construction below. Our PHFE scheme  $\text{PHFE}^{N,S} = (\text{PHFE.Setup}, \text{PHFE.KeyGen}, \text{PHFE.Enc}, \text{PHFE.Dec})$  for computing degree 3 multilinear polynomials  $g(\mathbf{x}, \mathbf{y}, \mathbf{z})$  in  $\mathcal{G}_\lambda^{N,S}$  proceeds as follows:

- $\text{PHFE.Setup}(1^\lambda, \text{pp})$  on input  $1^\lambda$  and public parameter  $\text{pp} = (p, G_1, G_2, G_T, \text{pair})$  samples a  $\mathbf{hIPE}$  master secret key  $\text{hmsk} \leftarrow \text{hIPE.Setup}(1^\lambda, \text{pp})$ , as well as two random vectors  $\mathbf{s}_1, \mathbf{s}_2 \leftarrow \mathbb{Z}_p^N$ . Output  $\text{msk} = (\text{hmsk}, \mathbf{s}^1, \mathbf{s}^2)$ .
- $\text{PHFE.KeyGen}(\text{msk}, g)$  on input a degree 3 multilinear polynomial  $g(\mathbf{x}, \mathbf{y}, \mathbf{z}) \in \mathcal{G}_\lambda^{N,S}$ , parses the computation of every output element  $g_l$  as

$$\forall l \in |g|, g_l(\mathbf{x}, \mathbf{y}, \mathbf{z}) = \mathbf{x}D_l(\mathbf{y} \otimes \mathbf{z}),$$

where  $D_l$  is a  $N \times N^2$  matrix over  $\mathbb{Z}_p$ . Then, for every  $l$ , generate a  $\mathbf{hIPE}$  ciphertext  $\text{hct}_l$  encrypting  $\bar{\mathbf{u}}_l = (D_l(\mathbf{s}_1 \otimes \mathbf{s}_2))||0$  using master secret key  $\text{hmsk}$ ,

$$\text{hct}_l \leftarrow \text{hIPE.Enc}(\text{hmsk}, \bar{\mathbf{u}}_l), \text{ where } \bar{\mathbf{u}}_l = (D_l(\mathbf{s}_1 \otimes \mathbf{s}_2))||0.$$

Output  $\text{sk} = \{\text{hct}_l, D_l\}_l$ .

- $\text{PHFE.Enc}(\text{msk}, \mathbf{x}, \mathbf{y}, \mathbf{z})$  on input vectors  $\mathbf{x}, \mathbf{y}, \mathbf{z} \in \mathbb{Z}_p^N$ . (Assume w.l.o.g. that every input vector contains a 1.) Do the following:
  - Sample a scalar  $r \leftarrow \mathbb{Z}_p$  and a  $\mathbf{hIPE}$  master secret key  $\text{hmsk} \leftarrow \text{hIPE.Setup}(1^\lambda, \text{pp})$ .
  - For every  $i \in [N]$ , generate a  $\mathbf{hIPE}$  ciphertext of vector  $\mathbf{u}_i = (rs_i^1||y_i||0)$ , and for every  $j \in [N]$ , generate a secret key of vector  $\mathbf{v}_j = (s_j^2||z_j||0)$ ,

$$\begin{aligned} \text{hct}_i &\leftarrow \text{hIPE.Enc}(\text{hmsk}, \mathbf{u}_i), \text{ where } \mathbf{u}_i = (rs_i^1||y_i||0), \\ \text{hsk}_j &\leftarrow \text{hIPE.KeyGen}(\text{hmsk}, \mathbf{v}_j), \text{ where } \mathbf{v}_j = (s_j^2||z_j||0). \end{aligned}$$

- Generate a  $\mathbf{hIPE}$  secret key of the vector  $\bar{\mathbf{v}} = (r\mathbf{x}||0)$ ,

$$\bar{\text{hsk}} \leftarrow \text{hIPE.KeyGen}(\text{hmsk}, \bar{\mathbf{v}}), \text{ where } \bar{\mathbf{v}} = (r\mathbf{x}||0).$$

Output  $\text{ct} = (\{\text{hct}_i, \text{hsk}_j\}_{i,j \in [N]}, \bar{\text{hsk}}, \mathbf{x})$ .

- $\text{PHFE.Dec}(\text{sk}, \text{ct})$  first decrypts all **hIPE** ciphertexts.

$$\begin{aligned} \forall i, j \in [N], \text{hIPE.Dec}(\text{hct}_i, \text{hsk}_j) &= [\langle \mathbf{u}_i, \mathbf{v}_j \rangle]_T = [rs_i^1 s_j^2 + y_i z_j]_T, \\ \forall l \in [|g|], \text{hIPE.Dec}(\bar{\text{hct}}_l, \bar{\text{hsk}}) &= [\langle \bar{\mathbf{u}}_l, \bar{\mathbf{v}} \rangle]_T = [r\mathbf{x}D_l(\mathbf{s}_1 \otimes \mathbf{s}_2)]_T. \end{aligned}$$

Concatenate the decrypted outputs of the first line for all  $i, j$ , which gives a one-time pad encryption of  $\mathbf{y} \otimes \mathbf{z}$  using key  $rs^1 \otimes s^2$ ,

$$\{\text{hIPE.Dec}(\text{hct}_i, \text{hsk}_j)\} = [rs^1 \otimes s^2 + \mathbf{y} \otimes \mathbf{z}]_T = \text{ict}.$$

Then, for every  $l \in |g|$ , the  $l$ 'th output element  $o_l$  can be recovered as

$$\langle \mathbf{x}D_l, \text{ict} \rangle - [r\mathbf{x}D_l(\mathbf{s}_1 \otimes \mathbf{s}_2)]_T = [\mathbf{x}D_l(\mathbf{y} \otimes \mathbf{z})]_T = [o_l]_T.$$

Output the concatenation  $[\mathbf{o}]_T$  of all  $[o_l]_T$ .

**Lemma 4.5.** *Assume the  $\mu$ -indistinguishability of the SXDH assumption over bilinear map groups. The above scheme  $\text{PHFE}^{N,S}$  is a partially-hiding functional encryption scheme for the class  $\mathcal{G}^{N,S}$  of degree 3 multilinear polynomials over  $\mathbb{Z}_p$ , satisfying correctness in the exponent, linear efficiency, and strong  $\text{poly}(\mu)$ -Sel-PH-Sim security.*

The above simulation security is strong because the simulator only takes an encoding of the output in  $G_1$ , instead of the output in the clear.

*Proof.* It is clear that the scheme is correct in the exponent and all algorithms run in time polynomial in  $\lambda, N, S$ . The encryption algorithm  $\text{PHFE.Enc}$  generates  $O(N)$  **hIPE** ciphertexts and secret keys of vectors of length 3, and a ciphertext of a vector of length  $O(N)$ , which by the efficiency of **hIPE**, takes  $\text{poly}(\lambda)N$  time. Therefore  $\text{PHFE}$  satisfies linear efficiency.

To show that  $\text{PHFE}$  satisfies strong  $\mu$ -Sel-PH-Sim-security, we build a sequence of hybrids  $H_0$  to  $H_{N+2}$  where the first hybrid  $H_0$  and  $H_n$  are respectively identical to **Real** and **Ideal** above.

**Hybrid  $H_1$ :** Recall that the  $\text{PHFE}$  secret key  $\text{sk}$  and ciphertext  $\text{ct}$  can be parsed as

$$\begin{aligned} \text{sk} &= \{\bar{\text{hct}}_l, D_l\}_l, \quad \text{ct} = (\{\text{hct}_i, \text{hsk}_j\}_{i,j \in [N]}, \bar{\text{hsk}}, \mathbf{x}), \\ \text{ct}^k &= (\{\text{hct}_i^k, \text{hsk}_j^k\}_{i,j \in [N]}, \bar{\text{hsk}}^k, \mathbf{x}^k), \end{aligned}$$

where  $\{\bar{\text{hct}}_l\}_l, \bar{\text{hsk}}$  and  $\{\bar{\text{hsk}}^k\}_k$  are all the **hIPE** ciphertexts and secret keys generated using master secret key  $\bar{\text{hmsk}}$ .

$H_1$  proceeds identically to  $H_0 = \text{Real}$  except that the **hIPE** ciphertexts  $\{\bar{\text{hct}}_l\}$  (in  $\text{sk}$ ) and secret key  $\bar{\text{hsk}}$  (in challenge ciphertext  $\text{ct}$ ) are changed to encode vectors:

$$\begin{aligned} \text{In } H_0, \quad \forall l, \quad \bar{\mathbf{u}}_l &= (D_l(\mathbf{s}^1 \otimes \mathbf{s}^2) \parallel 0) & \bar{\mathbf{v}} &= (r\mathbf{x} \parallel 0), \\ \text{In } H_1, \quad \forall l, \quad \bar{\mathbf{u}}_l &= (D_l(\mathbf{s}^1 \otimes \mathbf{s}^2) \parallel r\mathbf{x}D_l(\mathbf{s}^1 \otimes \mathbf{s}^2)) & \bar{\mathbf{v}} &= (\mathbf{0} \parallel 1). \end{aligned}$$

Secret keys  $\{\bar{\text{hsk}}^k\}$  in other ciphertexts  $\text{ct}^k$  do not change and remain encoded

$$\forall k, \quad \bar{\mathbf{v}}^k = (r^k \mathbf{x}^k \parallel 0).$$

Observe that all inner products between  $\{\bar{\mathbf{u}}_l\}$  and  $\bar{\mathbf{v}}, \{\bar{\mathbf{v}}^k\}$  remain the same. Therefore, it follows from the function hiding property of **HIPE** that  $H_0$  and  $H_1$  are indistinguishable.

We will switch the vector  $\mathbf{v}_j$  encoded in  $\text{hsk}_j$  from  $(s_j^2 || z_j || 0)$  to  $(s_j^2 || 0 || 0)$  one by one in the following sequence of hybrids  $\{H_{m+1:1}, H_{m+1:2}, H_{m+1:3}, H_{m+1:4}, H_{m+1:5}\}_{m \in [N]}$ .

**Hybrids  $H_{m+1}$  for  $m \in [N]$ :** This hybrid proceed identically to  $H_1$  except that the vectors  $\{\mathbf{v}_j\}_{j \leq m}$  encoded in the first  $m$  secret keys  $\{\text{hsk}_j\}_{j \leq m}$  are changed to encode  $\{(s_j^2 || 0 || 0)\}_{j \leq m}$  instead of  $\{(s_j^2 || z_j || 0)\}_{j \leq m}$ . This is done via the following 7 sub-hybrids.

**Hybrid  $H_{m+1:1}$ :** Toward changing  $\mathbf{v}_m$  from  $(s_m^2 || z_m || 0)$  to  $(s_m^2 || 0 || 0)$ , this hybrid changes the ciphertexts and secret keys  $\{\text{hct}_i^*, \text{hsk}_j^*\}_j$  generated using the master secret key  $\text{hmsk}^*$  (recall that **PHFE.Enc** samples a fresh master secret key for every ciphertext). When  $\star$  is empty, these keys and ciphertexts are contained in the challenge ciphertext  $\text{ct}$ , and when  $\star = k \in [t]$ , they are contained in ciphertexts  $\text{ct}^k$ . They are changed from encoding the top vectors in  $H_{m:7}$  ( $H_{1:7} = H_1$ ) to the bottom vectors in  $H_{m+1:1}$ .

$$\begin{aligned} \text{In } H_{m:5}, \quad \forall i, \star, \mathbf{u}_i^* &= (r^* s_i^1 || y_i^* || 0) & \mathbf{v}_m^* &= (r^* s_m^2 || z_m^* || 0) \\ \text{In } H_{m+1:1}, \quad \forall i, \star, \mathbf{u}_i^* &= (r^* s_i^1 || y_i^* || r^* s_i^1 s_m^2 + y_i^* z_m^*) & \mathbf{v}_m^* &= (0 || 0 || 1) \end{aligned}$$

All vectors  $\mathbf{v}_j, \mathbf{v}_j^k$  for  $j \neq m$  do not change and remain as follows:

$$\forall j < m, \star, \mathbf{v}_j^* = (r^* s_j^2 || 0 || 0) \quad \forall j > m, \star, \mathbf{v}_j^* = (r^* s_j^2 || z_j^* || 0).$$

Note that for every  $\star$ , the inner products between  $\mathbf{u}_i^*$  and  $\mathbf{v}_j^*$  for all  $i, j$  are identical in  $H_{m:5}$  and  $H_{m+1:1}$ . Applying the function hiding property of **HIPE** for every  $\star$  implies that  $H_{m:5}$  and  $H_{m+1:1}$  are indistinguishable.

**Hybrid  $H_{m+1:2}$ :** In this hybrid, for every  $i \in [N]$  we replace  $s_i^1 s_m^2$  with a random scalar  $t_i \leftarrow \mathbb{Z}_p$ . This means,

$$\begin{aligned} \text{In } H_{m+1:1}, \quad \forall i, \star, \mathbf{u}_i^* &= (r^* s_i^1 || y_i^* || r^* s_i^1 s_m^2 + y_i^* z_m^*) \\ &\forall l, \quad \bar{\mathbf{u}}_l = (D_l(\mathbf{s}^1 \otimes \mathbf{s}^2) || (r \mathbf{x} D_l(\mathbf{s}^1 \otimes \mathbf{s}^2) + \Delta_{l,m-1})) \\ \text{In } H_{m+1:2}, \quad \forall i, \star, \mathbf{u}_i^* &= (r^* s_i^1 || y_i^* || r^* t_i + y_i^* z_m^*) \\ &\forall l, \quad \bar{\mathbf{u}}_l = (D_l \mathbf{h}_m || (r \mathbf{x} D_l \mathbf{h}_m + \Delta_{l,m-1})), \\ &\quad \text{where } \mathbf{h}_m = \mathbf{s}^1 \otimes \mathbf{s}_{<m}^2 || \mathbf{t} || \mathbf{s}^1 \otimes \mathbf{s}_{>m}^2 \end{aligned}$$

and  $\mathbf{s}_{<m}^2 = s_1^2 \cdots s_{m-1}^2$ ,  $\mathbf{s}_{>m}^2 = s_{m+1}^2 \cdots s_N^2$ , and  $\mathbf{t} = t_1 \cdots t_N$ . (Ignore for the moment the scalar  $\Delta_{l,m-1}$ , which will be explained shortly in the next hybrid.) Note that  $\{s_i^1 s_m^2\}$  and  $s_m^2$  do not appear in any  $\mathbf{v}, \bar{\mathbf{v}}$  vectors

$$\forall \star, \mathbf{v}_j^* = \begin{cases} (r^* s_j^2 || 0 || 0) & j < m \\ (0 || 0 || 1) & j = m \\ (r^* s_j^2 || z_j^* || 0) & j > m \end{cases} \quad \bar{\mathbf{v}} = (\mathbf{0} || 1) \quad \forall k, \bar{\mathbf{v}}^k = (r^k \mathbf{x}^k || 0).$$

Since all  $\mathbf{u}, \bar{\mathbf{u}}$  vectors are encoded in ciphertexts of **HIPE**, by the canonical form of **HIPE** (see [Theorem 4.4](#)), these ciphertexts consist of encodings  $\{[L_1(\mathbf{u}_i^*)]_1\}$ ,  $\{[L_1(\bar{\mathbf{u}}_l)]_1\}$  in the first source group  $G_1$  of elements that are *linear* in these  $\mathbf{u}, \bar{\mathbf{u}}$  vectors. This means  $H_{m+1:1}$  can be perfectly emulated from  $\{[s_i^1]_1, [s_i^1 s_m^2]_1\}_i$  (by

sampling all other elements internally), while  $H_{m+1:2}$  can be perfectly emulated given  $\{[s_i^1]_1, [t_i]_1\}_i$ . It follows from the SXDH assumption w.r.t.  $G_1$  that  $H_{m+1:1}$  and  $H_{m+1:2}$  are indistinguishable.

**Hybrid  $H_{m+1:3}$ :** Similar to above, in this hybrid, for every  $i \in [N]$ , we replace  $rt_i$  with a random scalar  $t'_i \leftarrow \mathbb{Z}_p$ . Note that  $rt_i$  only appears in  $\mathbf{u}_i$  (for  $\star$  being empty) and  $\bar{\mathbf{u}}_l$ . Thus,

$$\begin{aligned} \text{In } H_{m+1:2}, \quad & \forall i, \quad \mathbf{u}_i = (rs_i^1 \parallel y_i \parallel rt_i + y_i z_m) \\ & \forall l, \quad \bar{\mathbf{u}}_l = ((\mathbf{w}_{\neq m} + D_{l,m} \mathbf{t}) \parallel (\mathbf{x}(r\mathbf{w}_{\neq m} + D_{l,m}(r\mathbf{t})) + \Delta_{l,m-1})), \\ & \quad \text{where } D_l = [D_{l,1} \mid \cdots \mid D_{l,m} \mid \cdots \mid D_{l,N}], \quad \mathbf{w}_{\neq m} = \sum_{i \neq m} D_{l,i}(\mathbf{s}^1 s_i^2) \\ \text{In } H_{m+1:3}, \quad & \forall i, \quad \mathbf{u}_i = (rs_i^1 \parallel y_i \parallel t'_i + y_i z_m) \\ & \forall l, \quad \bar{\mathbf{u}}_l = ((\mathbf{w}_{\neq m} + D_{l,m} \mathbf{t}) \parallel (\mathbf{x}(r\mathbf{w}_{\neq m} + D_{l,m} \mathbf{t}') + \Delta_{l,m-1})) \end{aligned}$$

It follows again from the canonical form of **hIPE** that  $H_{m+1:2}$  can be perfectly emulated from  $[r]_1, [\mathbf{t}]_1, [r\mathbf{t}]_1$  (by sampling all other elements internally), while  $H_{m+1:3}$  can be perfectly emulated from  $[r]_1, [\mathbf{t}]_1, [\mathbf{t}']_1$ . It follows from the SXDH assumption w.r.t.  $G_1$  that  $H_{m+1:2}$  and  $H_{m+1:3}$  are indistinguishable.

Observe that in this hybrid,  $\{\mathbf{u}_i\}$  contain  $\{t'_i + y_i z_m\}$ , which is a one-time pad encryption of  $y_i z_m$ . Therefore, we can define  $\mathbf{t}'' = \mathbf{t}' + \mathbf{y} z_m$  which is randomly distributed, and rewrite as follows:

$$\begin{aligned} \text{In } H_{m+1:3}, \quad & \forall i, \quad \mathbf{u}_i = (rs_i^1 \parallel y_i \parallel t''_i) \\ & \forall l, \quad \bar{\mathbf{u}}_l = ((\mathbf{w}_{\neq m} + D_{l,m} \mathbf{t}) \parallel (\mathbf{x}(r\mathbf{w}_{\neq m} + D_{l,m}(\mathbf{t}'' - \mathbf{y} z_m)) + \Delta_{l,m-1})) \end{aligned}$$

**Hybrid  $H_{m+1:4}$ :** In this hybrid, we change  $\mathbf{y} z_m$  to the zero vector  $\mathbf{0}$ , and  $\Delta_{l,m-1}$  to  $\Delta_{l,m} = \Delta_{l,m-1} - \mathbf{x} D_{l,m}(\mathbf{y} z_m)$ . That is,

$$\begin{aligned} \text{In } H_{m+1:4}, \quad & \forall i, \quad \mathbf{u}_i = (rs_i^1 \parallel y_i \parallel t''_i) \\ & \forall l, \quad \bar{\mathbf{u}}_l = ((\mathbf{w}_{\neq m} + D_{l,m} \mathbf{t}) \parallel (\mathbf{x}(r\mathbf{w}_{\neq m} + D_{l,m}(\mathbf{t}'' - \mathbf{0})) + \Delta_{l,m})). \end{aligned}$$

Since  $\mathbf{x} D_{l,m}(\mathbf{t}'' - \mathbf{0}) + \Delta_{l,m} = \mathbf{x} D_{l,m}(\mathbf{t}'' - \mathbf{y} z_m) + \Delta_{l,m-1}$ , hybrid  $H_{m+1:4}$  is identically distributed as  $H_{m+1:3}$ .

**Hybrid  $H_{m+1:5}$ :** In this hybrid, we reverse the change done in  $H_{m+1:3}$  — that is, for every  $i \in [N]$ , we replace  $t''_i$  with  $rt_i$ .

$$\begin{aligned} \text{In } H_{m+1:5}, \quad & \forall i, \quad \mathbf{u}_i = (rs_i^1 \parallel y_i \parallel rt_i) \\ & \forall l, \quad \bar{\mathbf{u}}_l = ((\mathbf{w}_{\neq m} + D_{l,m} \mathbf{t}) \parallel (\mathbf{x}(r\mathbf{w}_{\neq m} + D_{l,m}(r\mathbf{t})) + \Delta_{l,m})). \end{aligned}$$

It follows again from the same argument as in  $H_{m+1:3}$  that by the canonical form of **hIPE** and the SXDH assumption in  $G_1$ ,  $H_{m+1:4}$  is indistinguishable to  $H_{m+1:5}$ .

**Hybrid  $H_{m+1:6}$ :** Observe that  $H_{m+1:5}$  is identical to  $H_{m+1:2}$ , except that  $\mathbf{y} z_m$  is replaced with  $\mathbf{0}$ , or equivalently  $z_m$  replaced with 0, and  $\Delta_{l,m-1}$  replaced with  $\Delta_{l,m}$ . Therefore we can reverse the change done in  $H_{m+1:2}$  — that is, for every  $i \in [N]$ , we replace  $t_i$  with  $s_i^1 s_m^2$ .

$$\begin{aligned} \text{In } H_{m+1:5}, \quad & \forall i, \star, \quad \mathbf{u}_i^* = (r^* s_i^1 \parallel y_i^* \parallel r^* t_i + y_i^* z_m^*), \text{ with } z_m = 0 \\ & \forall l, \quad \bar{\mathbf{u}}_l = (D_l \mathbf{h}_m \parallel (r \mathbf{x} D_l \mathbf{h}_m + \Delta_{l,m})), \\ & \quad \text{where } \mathbf{h}_m = \mathbf{s}^1 \otimes \mathbf{s}_{< m}^2 \parallel \mathbf{t} \parallel \mathbf{s}^1 \otimes \mathbf{s}_{> m}^2. \\ \text{In } H_{m+1:6}, \quad & \forall i, \star, \quad \mathbf{u}_i^* = (r^* s_i^1 \parallel y_i^* \parallel r^* s_i^1 s_m^2 + y_i^* z_m^*), \text{ with } z_m = 0 \\ & \forall l, \quad \bar{\mathbf{u}}_l = (D_l(\mathbf{s}^1 \otimes \mathbf{s}^2) \parallel (r \mathbf{x} D_l(\mathbf{s}^1 \otimes \mathbf{s}^2) + \Delta_{l,m})). \end{aligned}$$

It follows again from the same argument as in  $H_{m+1:2}$  that by the canonical form of **hIPE** and the SXDH assumption in  $G_1$ ,  $H_{m+1:5}$  is indistinguishable to  $H_{m+1:6}$ .

**Hybrid  $H_{m+1:7}$ :** In this hybrid, we reverse the change done in  $H_{m+1:1}$ , that is,

$$\begin{aligned} \text{In } H_{m+1:6}, \quad \forall i, \star, \quad & \mathbf{u}_i^* = (r^* s_i^1 \parallel y_i^* \parallel r^* s_i^1 s_m^2 + y_i^* z_m^*) , \text{ with } z_m = 0, \\ & \mathbf{v}_m^* = (0 \parallel 0 \parallel 1). \\ \text{In } H_{m+1:7}, \quad \forall i, \star, \quad & \mathbf{u}_i^* = (r^* s_i^1 \parallel y_i^* \parallel 0), \\ & \mathbf{v}_m^* = (r^* s_m^2 \parallel z_m^* \parallel 0) , \text{ with } z_m = 0. \end{aligned}$$

It follows from the same argument as in  $H_{m+1:1}$  that by the function hiding of **hIPE**, hybrids  $H_{m+1:6}$  and  $H_{m+1:7}$  are indistinguishable.

Combining the above sub-hybrids, we observe that  $H_{m+1}$  changes  $z_m$  to 0 and  $\Delta_{l,m-1}$  to  $\Delta_{l,m}$ . Next, we summarize the last hybrid  $H_{N+1}$ .

Hybrid  $H_{N+1}$ : In this hybrid, all  $\{z_j\}$  values are changed to 0 and the value  $\Delta_{l,N}$  is used, that is,

$$\begin{aligned} \text{In } H_{N+1}, \quad \forall i, \star, \quad & \mathbf{u}_i^* = (r^* s_i^1 \parallel y_i^* \parallel 0) \quad \mathbf{v}_j^* = \begin{cases} (r s_j^2 \parallel 0 \parallel 0) & \star \text{ empty} \\ (r^k s_j^2 \parallel z_j^k \parallel 0) & \star = k \\ \forall l, \quad \bar{\mathbf{u}}_l = (D_l(\mathbf{s}^1 \otimes \mathbf{s}^2) \parallel (r \mathbf{x} D_l(\mathbf{s}^1 \otimes \mathbf{s}^2) + \Delta_N)). \end{cases} \end{aligned}$$

We now analyze the  $\Delta_{l,N}$  values: Since  $\Delta_0 = 0$  and  $\Delta_{l,m} = \Delta_{l,m-1} - \mathbf{x} D_{l,m}(\mathbf{y} z_m)$ ,

$$\Delta_{l,N} = \sum_{i \in [N]} \mathbf{x} D_{l,i}(\mathbf{y} z_i) = \mathbf{x} D_l(\mathbf{y} \otimes \mathbf{z}) = g_l(\mathbf{x}, \mathbf{y}, \mathbf{z}) .$$

In other words, the outputs  $\{g_l(\mathbf{x}, \mathbf{y}, \mathbf{z})\}$  are hardcoded in  $\bar{\mathbf{u}}_l$ , which are in turn encrypted using **hIPE** in  $\bar{\text{ct}}_l$  contained in the PHFE secret key  $\text{sk}$ .

**Hybrid  $H_{N+2}$  :** In this hybrid, for every  $i \in [N]$ , we change  $y_i$  to 0, that is,

$$\begin{aligned} \text{In } H_{N+1}, \quad \forall i, \quad & \mathbf{u}_i = (r s_i^1 \parallel y_i \parallel 0) \quad \mathbf{v}_j = (r s_j^2 \parallel 0 \parallel 0). \\ \text{In } H_{N+2}, \quad \forall i, \quad & \mathbf{u}_i = (r s_i^1 \parallel 0 \parallel 0) \quad \mathbf{v}_j = (r s_j^2 \parallel 0 \parallel 0). \end{aligned}$$

Recall that  $\{\mathbf{u}_i\}$  and  $\{\mathbf{v}_j\}$  are encoded, respectively, in ciphertexts and secret keys of **hIPE** using a freshly sampled master secret key  $\text{hmsk}$  at time encrypting the challenge message  $(\mathbf{x}, \mathbf{y}, \mathbf{z})$ . Since all inner products  $\{\langle \mathbf{u}_i, \mathbf{v}_j \rangle\}_{ij}$  are identical in  $H_{N+1}$  and  $H_{N+2}$ , it follows from the function hiding property of **hIPE** that these two hybrids are indistinguishable.

Observe that  $\mathbf{y}$  and  $\mathbf{z}$  are not used for generating  $H_{N+2}$  and the output  $g(\mathbf{x}, \mathbf{y}, \mathbf{z})$  is hardcoded in  $\{\bar{\mathbf{u}}_l\}$ , which are encrypted using **hIPE** in  $\{\bar{\text{ct}}_l\}$  contained in the PHFE secret key  $\text{sk}$ . By the canonical form of **hIPE**,  $\{\bar{\text{ct}}_l\}$  can thus be simulated from encodings  $[g(\mathbf{x}, \mathbf{y}, \mathbf{z})]_1$ . Thus,  $H_{N+2}$  can be generated by a simulator  $\text{PSim}$  with the following inputs as desired:

$$H_{N+2} = \text{PSim} \left( g, [g(\mathbf{x}, \mathbf{y}, \mathbf{z})]_1, \mathbf{x}, \left\{ \mathbf{x}^k, \mathbf{y}^k, \mathbf{z}^k \right\}_{i \in [t]} \right) .$$

Since there are  $O(N)$  hybrids in total, and all hybrids are  $\text{poly}(\mu)$ -indistinguishable, we conclude that PHFE satisfies  $\text{poly}(\mu)$ -Sel-PH-Sim security.  $\square$

## 4.2 PHFE for Polynomials with Polynomial Degree

Building upon the construction of PHFE for degree 3 multilinear polynomials  $g(\mathbf{x}, \mathbf{y}, \mathbf{z})$ , we attempt to construct PHFE for the more general case of  $g$  with *polynomial* degree in the public input  $\mathbf{x}$ , but still *linear* in the private inputs  $\mathbf{y}, \mathbf{z}$ , using again only bilinear maps. However, we can only handle a sub-class of such polynomials of the form  $g(\mathbf{x}, \mathbf{y}, \mathbf{z}) = q(f(\mathbf{x}), \mathbf{y}, \mathbf{z})$ , which first performs a complex high-degree computation  $f$  on  $\mathbf{x}$ , followed by a simple computation  $q$  on the output of the first step and  $\mathbf{y}, \mathbf{z}$ . Furthermore, we need  $q$  and  $f$  to satisfy certain size constraints — the length of the outputs of  $q/g$  is bounded by  $N^2$ , as well as the width of the *formula* that computes  $f$ . Below, we first formally define the subclass of polynomials considered and then give a construction of PHFE for computing them.

**The special class  $\mathcal{G}_\lambda^{N,S,\text{Dep}}$  of polynomials.** Fix an arbitrary modulus  $p(\lambda)$ . The function class  $\{\mathcal{G}_\lambda^{N,S,\text{Dep}}\}$  indexed by polynomials  $N, S$  and a logarithmic function  $\text{Dep}$  contains functions  $g$ , mapping three input vectors  $\mathbf{x}, \mathbf{y}, \mathbf{z} \in \mathbb{Z}_p^N$  to an output vector  $\mathbf{o} \in \mathbb{Z}_p^{|\mathbf{g}|}$ , that can be written in the following canonical form and satisfy the following constraints.

**Canonical Form:**  $g$  is decomposed into functions  $\alpha, \beta, f$  and  $\ell$  that depend either only on the public input  $\mathbf{x}$  or only on the private inputs  $\mathbf{y}, \mathbf{z}$ . More specifically, every output element  $g_i$  is computed as

$$\forall i \in [|\mathbf{g}|], \quad \begin{aligned} g_i(\mathbf{x}, \mathbf{y}, \mathbf{z}) &= \langle \alpha[i](\mathbf{y}, \mathbf{z}), f[i](\ell(\mathbf{x})) \rangle + \beta_i(\mathbf{y}, \mathbf{z}), \\ \alpha &= \alpha[1] || \cdots || \alpha[|\mathbf{g}|] \text{ and } f = f[1] || \cdots || f[|\mathbf{g}|], \end{aligned} \quad (3)$$

where  $\alpha[i]$  and  $f[i]$  denote the  $i$ 'th chunk of output elements of  $\alpha$  and  $f$  of appropriate equal length, and  $\beta_i$  the  $i$ 'th output element of  $\beta$ .  $f$  depends only on  $\ell(\mathbf{x})$  which is *linear* in the public input  $\mathbf{x}$ . We note that for different  $i$ , the fan-in of the inner product<sup>8</sup>, that is  $|f[i]| = |\alpha[i]|$ , may be different and is specified implicitly by  $g$ . For simplicity, we represent the computation of all output elements as

$$g(\mathbf{x}, \mathbf{y}, \mathbf{z}) = \langle \alpha(\mathbf{y}, \mathbf{z}), f(\ell(\mathbf{x})) \rangle + \beta(\mathbf{y}, \mathbf{z}).$$

Furthermore,  $f$  is a *formula* (i.e., all gates have fan-out 1) composed of alternating layers of *fan-in 2 multiplication* gates and layers of *unbounded fan-in addition* gates. Equivalently,  $f$  is composed of layers of *unbounded fan-in inner product* gates. We associate with this computation a binary tree  $T_f$ , where the root represents  $f$ , and every other node  $\gamma$  represents a function  $f^\gamma$  s.t.

$$f^\gamma = \langle f^{\gamma^0}, f^{\gamma^1} \rangle, \quad \text{i.e., } \forall i \in |f^\gamma|, \quad f_i^\gamma = \langle f^{\gamma^0}[i], f^{\gamma^1}[i] \rangle, \quad (4)$$

where we suppress the input  $\ell(\mathbf{x})$ .

We define the depth of  $f$  to be the depth of  $T_f$ , that is, the number of layers of inner products, and the size of a function  $X$  (e.g.,  $g, f, \alpha, \beta$ ) to be the number of fan-in two addition and multiplication over  $\mathbb{Z}_p$  needed for computing it.

We remark that any function  $g$  can be written in the above canonical form, for instance, by stratifying according to monomials in  $\mathbf{y}, \mathbf{z}$ , and writing the public computation on  $\mathbf{x}$  in formula. There might be multiple ways of writing a computation in canonical form, and some of them, perhaps all of them, are not efficient.

<sup>8</sup>The fan-in of the inner product refers to the length of the vectors in the computation.

**Constraints:** Here, we only consider  $g$  with canonical form satisfying the following constraints.

1.  $g$ , as well as  $\alpha, \beta$ , is linear in both  $\mathbf{y}$  and  $\mathbf{z}$  (its degree in  $\mathbf{x}$  may be an arbitrary polynomial).
2.  $g$  in its canonical form has polynomial size  $S$ .
3.  $f$  in the canonical form of  $g$  has logarithmic depth  $\text{Dep} = O(\log \lambda)$ .
4. The output length  $|g|$  of  $g$  and the *width*  $\text{width}(f)$  of  $f$  is bounded by  $N^2$ .

**Overview of our PHFE for  $\mathcal{G}^{N,S,\text{Dep}}$**  We give a recursive construction of PHFE for  $\mathcal{G}^{N,S,\text{Dep}}$ , in the depth  $\text{Dep}$  of the public computation  $f$ . In the base case, when  $\text{Dep} = 0$ , the function  $g$  is indeed degree 3 multi-linear and hence can be computed using the degree 3 PHFE constructed in the previous section. Next, assume that we have constructions of PHFE schemes  $\text{PHFE}^{\text{Dep}-1}$  for function classes  $\mathcal{G}^{N',S',\text{Dep}-1}$  with depth  $\text{Dep} - 1$ , we construct  $\text{PHFE}^{\text{Dep}}$  for  $\mathcal{G}^{N,S,\text{Dep}}$  with depth  $\text{Dep}$  using the following ideas.

Given a function  $g \in \mathcal{G}^{N,S,\text{Dep}}$  in the canonical form, every output element  $g_i$  can be computed as in [equation \(3\)](#). Let  $i_1, \dots, i_k$  be the indexes of elements in the  $i$ 'th output chunks of  $\alpha$  and  $f$ , that is,  $\alpha[i] = \alpha_{i_1, \dots, i_k}$  and  $f[i] = f_{i_1, \dots, i_k}$ ; we have

$$g_i = \sum_{j \in [k]} \alpha_{i_j} f_{i_j} + \beta_i,$$

where we suppress the input dependency for simplicity. Moreover, since every output element  $f_{i_j}$  of  $f$  is computed by an inner product  $\langle f^0[i_j], f^1[i_j] \rangle$  as in [equation \(4\)](#), we have

$$g_i = \langle \alpha_{i_1} f^0[i_1] \parallel \dots \parallel \alpha_{i_k} f^0[i_k] \parallel \beta_i, f^1[i_1] \parallel \dots \parallel f^1[i_k] \parallel 1 \rangle.$$

Note that the input vectors of the above inner product have public computation  $f^0, f^1$  of depth exactly  $\text{Dep} - 1$ . Thus, the key idea is using  $\text{PHFE}^{\text{Dep}-1}$  to compute the input vector. But clearly we cannot output the first input vector in the clear, which contains output elements of  $\alpha$  and  $\beta$  that depend on the private inputs  $\mathbf{y}, \mathbf{z}$ . Instead, we will use  $\text{PHFE}^{\text{Dep}-1}$  to compute the first input vector one-time-padded with a random vector  $\mathbf{t}[i]$ , together with the inner product of  $\mathbf{t}[i]$  and the second vector, that is,

$$\begin{aligned} g_i^0 &= (\alpha_{i_1} f^0[i_1] \parallel \dots \parallel \alpha_{i_k} f^0[i_k] \parallel \beta_i) + \mathbf{t}[i], \\ g_i^1 &= \langle \mathbf{t}[i], (f^1[i_1] \parallel \dots \parallel f^1[i_k] \parallel 1) \rangle. \end{aligned}$$

Given  $g_i^0, g_i^1$ , one can recover  $g_i$  as

$$g_i = \langle g_i^0[i], (f^1[i_1] \parallel \dots \parallel f^1[i_k] \parallel 1) \rangle - g_i^1. \quad (5)$$

Thanks to the random pad  $\mathbf{t}[i]$ ,  $g_i^0, g_i^1$  reveal only  $g_i$  and  $f^1$ . In summary, we reduce the computation of  $g$  to that of  $g^0 = \{g_i^0\}_i$  and  $g^1 = \{g_i^1\}_i$ .

To compute  $g^0, g^1$  using  $\text{PHFE}^{\text{Dep}-1}$ , it remains to argue that they satisfy the constraints of  $\mathcal{G}^{N',S',\text{Dep}-1}$  w.r.t. some (different)  $N', S'$ . Observe that the private computation of  $g^0, g^1$  involves computing  $\alpha, \beta$  on  $\mathbf{y}, \mathbf{z}$  and outputting the random pad  $\mathbf{t}$ , which is linear in  $\mathbf{y}, (\mathbf{z} \parallel \mathbf{t})$  (constraint 1). By setting  $N' = N + |\mathbf{t}| \geq 2N$ , we have that the public computation of  $g^0$  and  $g^1$ , which are exactly  $f^0$  and  $f^1$  on input  $\ell(\mathbf{x})$ , has depths  $\text{Dep} - 1$  and widths bounded

by  $\text{width}(f) \leq N^2 < N'^2$  (constraint 3 and 4). Their output lengths are both bounded by  $2N^2 < N'^2$  (constraint 4) since

$$|g^0| = \sum_{i \in [g]} (|f^0[i_1]| + \dots + |f^0[i_k]| + 1) = |f^0| + |g|, \quad |g^1| = |g|, \quad (6)$$

where the second last equality follows from that every  $f^0[i_j]$  is used only once (as  $f$  is a formula and each  $f_i$  is used only once). Finally, their sizes are polynomial (constraint 2)

$$\begin{aligned} \text{size}(g^0) &= O(\text{size}(f^0) + \text{size}(\alpha) + \text{size}(\beta)) = O(\text{size}(g)), \\ \text{size}(g^1) &= O(\text{size}(f^1) + |g|) = O(\text{size}(f^1) + \text{size}(\beta)) = O(\text{size}(g)). \end{aligned} \quad (7)$$

Therefore,  $\text{PHFE}^{\text{Dep}}$  recursively invokes  $\text{PHFE}^{\text{Dep}-1}$  to compute  $g^0, g^1$  and recovers  $g$  by [equation \(5\)](#). Its correctness and PHFE simulation security follow from that of  $\text{PHFE}^{\text{Dep}-1}$ .

Arguing linear efficiency is however a bit tricky. To ensure the linear efficiency of  $\text{PHFE}^{\text{Dep}}$ , we need the total length of the private inputs  $\mathbf{y}, \mathbf{z} || \mathbf{t}$  be bounded by  $O(N)$ . If so, after  $\text{Dep}$  levels of recursion, the length of private inputs grows to  $\text{poly}(\lambda)N$ , and the linear efficiency of the degree 3 PHFE scheme constructed in the previous section implies that of  $\text{PHFE}^{\text{Dep}}$ . However,  $\mathbf{t}$  is a one-time pad for  $g^0$  and  $|vt| = |g^0| = |f^0| + |g|$  (see [equation \(6\)](#)). Restricting  $|\mathbf{t}| = O(N)$  means  $\text{PHFE}^{\text{Dep}}$  can only compute functions with linear-length outputs, which is insufficient for our application later of evaluating pseudo-flawed smudging generators with polynomial  $N^{1+\varepsilon}$  length outputs. To circumvent this problem, instead of sampling  $\mathbf{t}$  randomly, we compute it as the tensor product  $\mathbf{t} = \mathbf{u}_0 \otimes \mathbf{u}_1$  of two random vectors of length- $\sqrt{|\mathbf{t}|}$ . We show that as long as  $|g|$  and  $\text{width}f$  are bounded by  $\text{poly}(\lambda)N^2$  (constraint 4), the length of  $\mathbf{t}$  used at *every* recursion level is bounded by  $\text{poly}(\lambda)N^2$  and hence the lengths of the actual private inputs  $|\mathbf{u}_0| = |\mathbf{u}_1|$  are always bounded by  $\text{poly}(\lambda)N$ . For security, to address the problem that  $\mathbf{t} = \mathbf{u}_0 \otimes \mathbf{u}_1$  is no longer random, we use the fact that in our construction  $\mathbf{t}$  is computed in the exponent of bilinear map groups and never revealed in the clear, and hence the SXDH assumption implies that  $\mathbf{t}$  is pseudo-random in the exponent.

**Construction of our PHFE for  $\mathcal{G}^{N,S,\text{Dep}}$**  Let  $\text{Dep}$  be a logarithmic function and  $N, S$  polynomials. Using a PHFE scheme  $\text{PHFE}^{\text{Dep}-1}$  for  $\mathcal{G}^{N',S',\text{Dep}-1}$  with appropriate  $N', S'$  set below, we construct a PHFE scheme  $\text{PHFE}^{N,S,\text{Dep}}$  for  $\mathcal{G}^{N,S,\text{Dep}}$  as follows:

- $\text{PHFE.Setup}^{\text{Dep}}(1^\lambda, \text{pp})$  on input  $1^\lambda$  and public parameter  $\text{pp} = (p, G_1, G_2, G_T, \text{pair})$  samples two  $\text{PHFE}^{\text{Dep}-1}$  master secret keys,

$$\forall b \in \{0, 1\}, \quad \text{msk}^b \leftarrow \text{PHFE.Setup}^{\text{Dep}-1}(1^\lambda, \text{pp}).$$

Output  $\text{msk} = (\text{msk}^0, \text{msk}^1)$ .

- $\text{PHFE.KeyGen}^{\text{Dep}}(\text{msk}, g)$  on input a function  $g \in \mathcal{G}^{N,S,\text{Dep}}$  of canonical form

$$\begin{aligned} \forall i \in [g], \quad g_i(\mathbf{x}, \mathbf{y}, \mathbf{z}) &= \langle \alpha[i](\mathbf{y}, \mathbf{z}), f[i](\ell(\mathbf{x})) \rangle + \beta_i(\mathbf{y}, \mathbf{z}), \text{ and} \\ \forall j \in [f], \quad f_j(\ell(\mathbf{x})) &= \langle f^0[j](\ell(\mathbf{x})), f^1[j](\ell(\mathbf{x})) \rangle, \end{aligned}$$

does the following:

- Prepare functions  $g^0, g^1$  with additional private inputs  $\mathbf{u}_0, \mathbf{u}_1 \in \mathbb{Z}_p^{2N}$  as follows,

$$\begin{aligned} \forall i \in [|g|], \quad & g^0[i](\mathbf{x}, (\mathbf{y}||\mathbf{u}_0), (\mathbf{z}||\mathbf{u}_1)) = (\alpha_{i_1} f^0[i_1] || \cdots || \alpha_{i_k} f^0[i_k] || \beta_i) + \mathbf{t}[i], \\ & g_i^1(\mathbf{x}, \mathbf{u}_0, \mathbf{u}_1) = \langle \mathbf{t}[i], (f^1[i_1] || \cdots || f^1[i_k] || 1) \rangle, \\ & \mathbf{t}(\mathbf{u}_0, \mathbf{u}_1) = \mathbf{u}_0 \otimes \mathbf{u}_1 = (\mathbf{t}[1] || \mathbf{t}[i] || \mathbf{t}[|g|] || \star), \end{aligned}$$

where  $i_1, \dots, i_k$  are indexes of output elements contained in chunk  $\alpha[i]$  and  $f[i]$  (specified implicitly by  $g$ ).<sup>9</sup> Note that the input lengths  $N'$  of  $g^0, g^1$  are bounded by  $3N$  and by equation (7), their sizes  $S'$  are bounded by  $O(S)$ .

- For every  $b \in \{0, 1\}$ , generate a  $\text{PHFE}^{\text{Dep}-1}$  secret key of  $g^b$  using  $\text{msk}^b$ ,

$$\text{sk}^b \leftarrow \text{PHFE.KeyGen}^{\text{Dep}-1}(\text{msk}^b, g^b).$$

Output  $\text{sk} = (\text{sk}^0, \text{sk}^1, g)$ .

- $\text{PHFE.Enc}^{\text{Dep}}(\text{msk}, \mathbf{x}, \mathbf{y}, \mathbf{z})$  does the following:

- Sample two random vectors of length  $2N$ ,  $\mathbf{u}_0, \mathbf{u}_1 \leftarrow \mathbb{Z}_p^{2N}$ .
- Generate two  $\text{PHFE}^{\text{Dep}-1}$  ciphertexts

$$\begin{aligned} \text{ct}^0 &\leftarrow \text{PHFE.Enc}^{\text{Dep}-1}(\text{msk}^0, \mathbf{x}, (\mathbf{y}||\mathbf{u}_0), (\mathbf{z}||\mathbf{u}_1)), \\ \text{ct}^1 &\leftarrow \text{PHFE.Enc}^{\text{Dep}-1}(\text{msk}^1, \mathbf{x}, \mathbf{u}_0, \mathbf{u}_1). \end{aligned}$$

Output  $\text{ct} = (\text{ct}^0, \text{ct}^1, \mathbf{x})$ .

- $\text{PHFE.Dec}^{\text{Dep}}(\text{sk}, \text{ct})$  first decrypts both  $\text{PHFE}^{\text{Dep}-1}$  ciphertexts

$$\forall b \in \{0, 1\}, \quad [\mathbf{o}^b]_T = \text{PHFE.Dec}^{\text{Dep}-1}(\text{sk}^b, \text{ct}^b).$$

According to  $g$ , parse  $[\mathbf{o}^0]_T = [\mathbf{o}^0[1]]_T || \cdots || [\mathbf{o}^0[|g|]]_T$ , where  $|\mathbf{o}^0[i]| = |g^0[i]|$ . For every  $i \in [|g|]$ , compute

$$[o_i]_T = \langle [\mathbf{o}^0[i]]_T, (f^1[i_1] || \cdots || f^1[i_k] || 1) \rangle - [o_i^1]_T,$$

where the output elements of  $f^1$  can be computed using the public input  $\mathbf{x}$ . Output the concatenation  $[\mathbf{o}]_T$  of all  $[o_i]_T$ .

*Correctness:* The correctness in exponent of  $\text{PHFE}^{\text{Dep}}$  follows immediately from that of  $\text{PHFE}^{\text{Dep}-1}$ . By the latter, we know that  $\mathbf{o}^0[i] = g^0[i](\mathbf{x}, \mathbf{y}||\mathbf{u}_0, \mathbf{z}||\mathbf{u}_1)$  and  $o_i^0 = g_i^1(\mathbf{x}, \mathbf{u}_0, \mathbf{u}_1)$ . It then follows from the definition of  $g^0$  and  $g^1$  that we have

$$\mathbf{o}_i = g_i = \langle g^0[i], (f^1[i_1] || \cdots || f^1[i_k] || 1) \rangle - g_i^1.$$

Thus the output of  $g_i$  is computed correctly in the exponent.

*Linear Efficiency:*  $\text{PHFE.Enc}^{\text{Dep}}$  makes two calls to  $\text{PHFE.Enc}^{\text{Dep}-1}$  with inputs of length  $3N$ . After all Dep levels of recursion, it makes at most  $2^{\text{Dep}}$  calls to  $\text{PHFE.Enc}^0$  with inputs of length

<sup>9</sup>As analyzed above in equation (6), since  $|\mathbf{t}| = |g^0| + |f| \leq 2N^2$ ,  $\mathbf{u}_0 \otimes \mathbf{u}_1$  is long enough to cover  $\mathbf{t}$ .

$3^{\text{Dep}}N$ . The base case scheme  $\text{PHFE.Enc}^0$  is the PHFE scheme for computing degree 3 multi-linear polynomials constructed in Section 4.1, which has linear efficiency. Hence the encryption time of  $\text{PHFE.Enc}^{\text{Dep}}$  is bounded by  $2^{\text{Dep}}O(3^{\text{Dep}}N)$ , which is bounded by  $\text{poly}(\lambda)N$  as  $\text{Dep}$  is logarithmic in  $\lambda$ . Since  $\text{PHFE.Dec}^{\text{Dep}-1}$  has linear efficiency,  $T^{\text{Dep}}(N) = 2\text{poly}(\lambda)O(3N) = \text{poly}(\lambda)N$ .

Strong PHFE Simulation Security: We show the following theorem:

**Theorem 4.6.** *Let  $p, N, S, \text{Dep}$  and bilinear map groups  $(\text{pp}, G_1, G_2, G_T, \text{pair})$  be defined above, and let  $\mu$  be any negligible function. If the  $\mu$ -indistinguishability of the SXDH assumption holds w.r.t.  $G_1$ , then  $\text{PHFE}^{\text{Dep}-1}$  above is a partially-hiding functional encryption for  $\mathcal{G}^{N, S, \text{Dep}}$ , with strong  $\text{poly}(\mu)$ -Sel-PH-Sim security.*

The above theorem follows from the security of the base scheme  $\text{PHFE}^0$  (Lemma 4.5) and the following lemma on the security loss due at every recursion level. Lemma 4.5 states that it follows from the  $\mu$ -indistinguishability of SXDH w.r.t.  $G_1$  that  $\text{PHFE}^0$  is strong  $\text{poly}(\mu)$ -Sel-PH-Sim-secure. By the following lemma, after  $\text{Dep}$  levels of recursion, we obtain  $\text{PHFE}^{\text{Dep}}$  with  $3^{\text{Dep}}\text{poly}(\mu)$ -Sel-Sim security. Since  $\text{Dep}$  is logarithmic and  $\mu$  is negligible,  $\text{PHFE}^{\text{Dep}}$  is  $\text{poly}(\mu)$ -Sel-Sim secure.

**Lemma 4.7.** *Let  $p, N, S, N', S', \text{Dep}$  and bilinear map groups  $(\text{pp}, G_1, G_2, G_T, \text{pair})$  be as defined above. If  $\text{PHFE}^{\text{Dep}-1}$  is a partially-hiding functional encryption scheme for  $\mathcal{G}^{N', S', \text{Dep}-1}$  with strong  $\mu$ -Sel-PH-Sim security, and the  $\mu \cdot \text{negl}$ -indistinguishability of the SXDH assumption holds w.r.t.  $G_1$ , then  $\text{PHFE}^{\text{Dep}}$  above is a partially-hiding functional encryption for  $\mathcal{G}^{N, S, \text{Dep}}$ , with strong  $3\mu$ -Sel-PH-Sim security.*

*Proof.* We want to show a PPT universal simulator  $\text{PSim}$ , such that, for every security parameter  $\lambda$ , bilinear map groups  $\text{pp} = (p, G_1, G_2, G_T, \text{pair})$ , function  $g \in \mathcal{G}_\lambda^{N, S, \text{Dep}}$ , and input vectors  $(\mathbf{x}, \mathbf{y}, \mathbf{z})$ ,  $\{\mathbf{x}^k, \mathbf{y}^k, \mathbf{z}^k\}_{k \in [t]}$  for some polynomial  $t = t(\lambda)$ , the following distributions are  $3\mu(\lambda)$ -indistinguishable:

$$\text{Real} = \left\{ \begin{array}{l} \text{msk} \leftarrow \text{PHFE.Setup}^{\text{Dep}}(1^\lambda, \text{pp}) \\ \text{sk} \leftarrow \text{PHFE.KeyGen}^{\text{Dep}}(\text{msk}, g) \\ \text{ct} \leftarrow \text{PHFE.Enc}^{\text{Dep}}(\text{msk}, \mathbf{x}, \mathbf{y}, \mathbf{z}) \\ \{\text{ct}^k \leftarrow \text{PHFE.Enc}^{\text{Dep}}(\text{msk}, \mathbf{x}^k, \mathbf{y}^k, \mathbf{z}^k)\}_{i \in [t]} \end{array} : \text{sk}, \text{ct}, \{\text{ct}^k\}_{i \in [t]} \right\},$$

$$\text{Ideal} = \left\{ \text{PSim} \left( g, [g(\mathbf{x}, \mathbf{y}, \mathbf{z})]_1, \mathbf{x}, \{\mathbf{x}^k, \mathbf{y}^k, \mathbf{z}^k\}_{i \in [t]} \right) \right\}.$$

We construct the simulator via a sequence of hybrids  $H_1$  to  $H_4$

**Hybrid  $H_1$ :** By construction, each  $\text{PHFE}^{\text{Dep}}$  secret key  $\text{sk}$  and ciphertext contains respectively two secret keys  $\text{sk}^0, \text{sk}^1$  and ciphertexts  $\text{ct}^0, \text{ct}^1$  of  $\text{PHFE}^{\text{Dep}-1}$ , generated using different master secret keys  $\text{msk}^0, \text{msk}^1$ . Each copy is used to compute the function  $g^0$  and  $g^1$  on inputs  $(\mathbf{x}, \mathbf{y} || \mathbf{u}_0, \mathbf{z} || \mathbf{u}_1)$  and  $(\mathbf{x}, \mathbf{u}_0, \mathbf{u}_1)$ , respectively. In this hybrid, we apply the strong PHFE simulation security of  $\text{PHFE}^{\text{Dep}-1}$  to simulate each set of keys and ciphertexts generated using the same master secret key.

In  $\text{Real}$ ,  $\text{sk} = (\text{sk}^0, \text{sk}^1)$ ,  $\text{ct} = (\text{ct}^0, \text{ct}^1)$ ,  $\{\text{ct}^k = (\text{ct}^{k,0}, \text{ct}^{k,1})\}^k$ .

In  $H_1$ ,  $\forall b \in \{0, 1\}$ ,  $(\text{sk}^b, \text{ct}^b, \{\text{ct}^{k,0}\}_k) \leftarrow \text{PSim}^{\text{Dep}-1} \left( g^b, [\mathbf{o}^b]_1, \mathbf{x}, \{\mathbf{x}^k, \mathbf{y}^k || \mathbf{u}_0^k, \mathbf{z}^k || \mathbf{u}_1^k\} \right)$ ,

$$\mathbf{o}^b = \begin{cases} g^0(\mathbf{x}, \mathbf{y} || \mathbf{u}_0, \mathbf{z} || \mathbf{u}_1), & b = 0, \\ g^1(\mathbf{x}, \mathbf{u}_0, \mathbf{u}_1), & b = 1, \end{cases}$$

where  $\mathbf{u}_b^*$  is a random sampled vector of length  $2N$ . It follows from the  $\mu$ -Sel-PH-Sim-security of  $\text{PHFE}^{\text{Dep-1}}$  that  $\text{Real}$  and  $H_1$  are  $2\mu$ -indistinguishable.

**Hybrid  $H_2$ :** By construction, for every  $i \in [|g|]$ , the output of  $g^0[i]$  is a one-time pad encryption using pad  $\mathbf{t}[i]$  and the output  $g_i^1$  is an inner product between  $\mathbf{t}[i]$  and vector  $(f^1[i_1] \parallel \cdots \parallel f^1[i_k] \parallel 1)$ , where  $\mathbf{t} = \mathbf{u}_0 \otimes \mathbf{t}_1$ . Thus both  $g^0, g^1$  are linear in  $\mathbf{t}$ . In this hybrid, we replace  $\mathbf{u}_0, \otimes \mathbf{u}_1$  with a freshly sampled random vector  $\mathbf{t} \leftarrow \mathbb{Z}_p^{4N^2}$  of the same length, and prepare  $\mathbf{o}^b$  used for simulation as follows:

$$\begin{aligned} \mathbf{o}^0[i] &= (\alpha_{i_1} f^0[i_1] \parallel \cdots \parallel \alpha_{i_k} f^0[i_k] \parallel \beta_i) + \mathbf{t}[i], \\ \mathbf{o}_i^1 &= \langle \mathbf{t}[i], (f^1[i_1] \parallel \cdots \parallel f^1[i_k] \parallel 1) \rangle. \end{aligned}$$

Observe that in both  $H_1$  and  $H_2$ , only encodings of  $\mathbf{o}^b$  in  $G_1$  are needed for simulation, thus given encodings  $[\mathbf{u}_0 \otimes \mathbf{u}_1]_1$  and  $[\mathbf{t}]_1$ , we can perfectly emulate the encodings  $[\mathbf{o}^b]_1$  needed for simulation in  $H_1, H_2$ . It thus follows from the  $\mu \cdot \text{negl}$ -indistinguishability of the SXDH assumption w.r.t.  $G_1$  that these two hybrids are  $\mu$ -indistinguishable.

**Hybrid  $H_3$ :** In hybrid  $H_2$ ,  $\mathbf{o}^0$  is distributed randomly as it is a one-time pad encryption using a truly random key  $\mathbf{t}$ , and  $\mathbf{o}^1$  satisfies that  $\mathbf{o}_i^1 = \langle \mathbf{o}^0[i], (f^1[i_1] \parallel \cdots \parallel f^1[i_k] \parallel 1) \rangle - \mathbf{o}$ , where  $\mathbf{o} = g(\mathbf{x}, \mathbf{y}, \mathbf{z})$ . Therefore, we can construct a simulator  $\text{PSim}^{\text{Dep}}$  that on input  $(g, [\mathbf{o}]_1, \mathbf{x}, \{\mathbf{x}^k, \mathbf{y}^k, \mathbf{z}^k\}_k)$ , samples  $\mathbf{o}^0, \{\mathbf{u}_b^k\}$  randomly, computes  $[\mathbf{o}^1]_1$  as above, and invokes  $\text{PSim}^{\text{Dep-1}}$  twice as in  $H_2$ . The distribution output by  $\text{PSim}^{\text{Dep}}$  is identical to that generated in  $H_2$ .

$H_3$  is exactly the distribution  $\text{Ideal}$  and by a hybrid argument is  $3\mu$ -indistinguishable to the distribution  $\text{Real}$ .  $\square$

## 5 Noisy Secret-Key Linear Functional Encryption

### 5.1 Definitions

In this section, we define  $\eta$ -noisy secret-key linear functional encryption, where  $\eta$  is a distribution over the codomain of the functions supported by the functional encryption scheme. For our construction in [Section 6](#),  $\eta$  is a linear combination of flawed-smudging distributions. We here only give definitions for linear functions since this is sufficient for our construction, but all definitions easily extend to general function classes.

Noisy secret-key FE schemes have the same syntax as regular secret-key FE schemes (cf. [Definition 2.16](#)), but decrypting an encryption of  $x$  with a secret key for the function  $f$  yields  $f(x) + e$  for  $e \leftarrow \eta$  (instead of  $f(x)$ ). We further only require weak correctness in the sense that decryption only needs to succeed if all coordinates of  $f(x)$  lie in a polynomially sized range  $\mathcal{R}$  known to the decryptor. This generalizes correctness in the exponent as defined in [Definition 4.2](#).

Noisy linear function encryption has recently been introduced by Agrawal [[Agr18a](#)], but our notion differs in three points: First, we require the decryption to recover  $f(x) + e$ , where  $e$  is from a specific distribution  $\eta$ , while Agrawal only requires  $e$  to have bounded norm. Secondly, we only require weak correctness. And thirdly, we consider simulation-security, whereas Agrawal defines indistinguishability-based security for noisy linear functional encryption.

**Definition 5.1** (Noisy secret-key FE). Let  $n = n(\lambda)$ ,  $m = m(\lambda)$ , and  $p = p(\lambda)$  be positive integers for  $\lambda \in \mathbb{N}$ . Further let  $\eta = \eta(\lambda)$  be a distribution over  $\mathbb{Z}_p^m$ , and let  $B = B(\lambda) \leq \text{poly}(\lambda)$  be a polynomial bound. An  $\eta$ -noisy secret-key linear functional encryption scheme NFE for functions  $\mathbb{Z}_p^n \rightarrow \mathbb{Z}_p^m$  consists of the following four PPT algorithms.

- *Setup*:  $\text{NFE.Setup}(1^\lambda)$  is an algorithm that on input a security parameter, outputs a master secret key  $\text{nmsk}$ .
- *Key Generation*:  $\text{NFE.KeyGen}(\text{nmsk}, f)$  on input the master secret key  $\text{nmsk}$  and the description of a linear function  $f: \mathbb{Z}_p^n \rightarrow \mathbb{Z}_p^m$ , outputs a secret key  $\text{nsk}_f$ .
- *Encryption*:  $\text{NFE.Enc}(\text{nmsk}, \mathbf{x})$  on input the master secret key  $\text{nmsk}$  and a message  $\mathbf{x} \in \mathbb{Z}_p^n$ , outputs an encryption  $\text{nct}$  of  $\mathbf{x}$ .
- *Decryption*:  $\text{NFE.Dec}(\text{nsk}_f, \text{nct}, \mathcal{R}_1, \dots, \mathcal{R}_m)$  on input a secret key  $\text{nsk}_f$ , a ciphertext  $\text{nct}$ , and polynomially sized sets  $\mathcal{R}_1, \dots, \mathcal{R}_m \subseteq \mathbb{Z}_p$ , outputs  $\mathbf{y} \in \mathbb{Z}_p^m$ .

**Weak correctness.** For all  $\lambda \in \mathbb{N}$ , for all linear functions  $f: \mathbb{Z}_p^n \rightarrow \mathbb{Z}_p^m$ , for all  $\mathbf{x} \in \mathbb{Z}_p^n$ , and for all sets  $\mathcal{R}_1, \dots, \mathcal{R}_m \subseteq \mathbb{Z}_p$ , we have: If  $|\mathcal{R}_i| \leq B$  for all  $i \in [m]$ , and if  $\lfloor f(\mathbf{x}) \rfloor_p \in \mathcal{R}_1 \times \dots \times \mathcal{R}_m$ , then

$$\Pr \left[ \begin{array}{l} \text{nmsk} \leftarrow \text{NFE.Setup}(1^\lambda) \\ \text{nsk}_f \leftarrow \text{NFE.KeyGen}(\text{nmsk}, f) \\ \text{nct} \leftarrow \text{NFE.Enc}(\text{nmsk}, \mathbf{x}) \\ \mathbf{y} \leftarrow \text{NFE.Dec}(\text{nsk}_f, \text{ct}, \mathcal{R}_1, \dots, \mathcal{R}_m) \end{array} : \exists \mathbf{e} \in \text{Support}(\eta) \ \mathbf{y} = \lfloor f(\mathbf{x}) + \mathbf{e} \rfloor_p \right] = 1.$$

Note that the correctness definition only requires  $\mathbf{e}$  to be in the support of  $\eta$ , not that it has distribution  $\eta$ . This is sufficient for correctness in our construction in Section 6, which uses NFE. The actual distribution  $\eta$  is important for security, which we define next.

**Simulation security.** We next define 1-key simulation-based fully-selective security for noisy FE schemes. Similarly to the corresponding definition for secret-key FE schemes, it requires the existence of a simulator that can simulate a secret key for a function  $f$  and a ciphertext for  $\mathbf{x}$  given only  $f$  and  $f(\mathbf{x}) + \mathbf{e}$ , where  $\mathbf{e}$  is sampled from  $\eta$ . Additionally, the simulator has to simulate further ciphertexts for other inputs  $\mathbf{x}_i$  given these  $\mathbf{x}_i$ . The latter is needed since in the secret-key setting, adversaries cannot produce ciphertexts on their own and simulated ciphertexts should still be indistinguishable from real ones if one sees additional ciphertexts. Fully-selective here means that the function and challenge input cannot be chosen adaptively, but are fixed at the beginning.

**Definition 5.2** (1-key Sel-Sim-security for noisy secret-key FE). Let  $\text{NFE} = (\text{NFE.Setup}, \text{NFE.KeyGen}, \text{NFE.Enc}, \text{NFE.Dec})$  be an  $\eta$ -noisy secret-key linear functional encryption scheme for functions  $\mathbb{Z}_p^n \rightarrow \mathbb{Z}_p^m$ . We say NFE is *1-key  $\mu$ -Sel-Sim-secure* if there is a PPT universal simulator  $\text{Sim}$  such that for every ensemble of linear functions  $\{f_\lambda\}_{\lambda \in \mathbb{N}}$ , where  $f_\lambda: \mathbb{Z}_{p(\lambda)}^{n(\lambda)} \rightarrow \mathbb{Z}_{p(\lambda)}^{m(\lambda)}$ , every ensemble of inputs  $\{\mathbf{x}_\lambda^*\}_{\lambda \in \mathbb{N}}$ , every polynomial  $t$ , and every ensemble of sequences of inputs

$\{\mathbf{x}_{i,\lambda}\}_{\lambda \in \mathbb{N}, i \in [t(\lambda)]}$ , where  $\mathbf{x}_\lambda^*, \mathbf{x}_{i,\lambda} \in \mathbb{Z}_{p(\lambda)}^{n(\lambda)}$ , the following distributions are  $\mu$ -indistinguishable:

$$\left\{ \begin{array}{l} \text{nmsk} \leftarrow \text{NFE.Setup}(1^\lambda) \\ \text{nsk}_f \leftarrow \text{NFE.KeyGen}(\text{nmsk}, f_\lambda) \\ \text{nct}^* \leftarrow \text{NFE.Enc}(\text{nmsk}, \mathbf{x}_\lambda^*) \\ \{\text{nct}_i \leftarrow \text{NFE.Enc}(\text{nmsk}, \mathbf{x}_{i,\lambda})\}_{i \in [t(\lambda)]} \end{array} : \text{nsk}_f, \text{nct}^*, \{\text{nct}_i\}_{i \in [t(\lambda)]} \right\}_{\lambda \in \mathbb{N}},$$

$$\left\{ \mathbf{e} \leftarrow \eta(\lambda) : \text{Sim}\left(f_\lambda, [f_\lambda(\mathbf{x}_\lambda^*) + \mathbf{e}]_p, \{\mathbf{x}_{i,\lambda}\}_{i \in [t(\lambda)]}\right) \right\}_{\lambda \in \mathbb{N}}.$$

## 5.2 Construction from PHFE and Noise Generator

There is a simple construction of an  $\eta$ -noisy secret-key linear FE scheme from a PHFE scheme for a function class  $\mathcal{G}$  and a noise generator  $G$  in the class  $\mathcal{G}$  with the following property: Seeds for the generator are split into a public and a private part, and the output distributions are indistinguishable from  $\eta$  even when given the public part of the seed. The NFE scheme encrypts a value  $\mathbf{x}$  by encrypting  $\mathbf{x}$  as part of the private input of PHFE, and encrypting a seed  $\phi$  for the noise generator split over the public and private inputs. To generate a key for a function  $f$ , it generates a key for the function  $g(\mathbf{x}, \phi) = f(\mathbf{x}) + G(\phi)$ . Decryption clearly recovers  $f(\mathbf{x}) + \mathbf{e}$  for some  $\mathbf{e}$  in the support of  $\eta$ , and security follows from simulation security of the FE scheme and the properties of  $G$ . To achieve sublinear compactness, we need  $G$  to have superlinear stretch.

We now proceed with the formal description of the used primitives and the construction. Let  $n = n(\lambda)$ , and  $m = m(\lambda)$  be polynomials, let  $p = p(\lambda)$  be super-polynomial, and let  $\mathcal{F} = \mathcal{F}(\lambda)$  be the set of linear functions  $\mathbb{Z}_p^n \rightarrow \mathbb{Z}_p^m$ . Further let  $\eta = \eta(\lambda)$  be a distribution over  $\mathbb{Z}^m$  with polynomially bounded support and let  $\alpha > 0$  be a constant. We construct an  $\eta$ -noisy secret-key FE scheme NFE for  $\mathcal{F}$  using the following tools:

- Partially-hiding functional encryption schemes  $\text{PHFE}^{n,m}$  for computing functions in a class  $\{\mathcal{G}_\lambda\}_{\lambda \in \mathbb{N}}$  with public domains  $\mathcal{X}_\lambda := \mathbb{Z}_p^{m^{1-\alpha}}$ , private domains  $\mathcal{Y}_\lambda := \mathbb{Z}_p^{m^{1-\alpha}} \times \mathbb{Z}_p^{n+m^{1-\alpha}}$ , and ranges  $\mathcal{Z}_\lambda := \mathbb{Z}_p^m$ . We assume the schemes satisfy correctness in the exponent as defined in [Definition 4.2](#), and linear efficiency, i.e., the encryption time is  $\text{poly}(\lambda)(n+m^{1-\alpha})$ , depending linearly on the input length and independent of the size of the computation. Furthermore, we assume that the scheme is 1-key  $O(\mu)$ -Sel-PH-Sim-secure.
- A family of distributions  $\{\Gamma_\lambda\}_{\lambda \in \mathbb{N}}$  with the same syntax and efficiency as PFGs, with seeds  $\phi$  divided into three parts  $\phi_1, \phi_2, \phi_3$  such that outputs are  $O(\mu)$ -indistinguishable from  $\eta$  even when given  $\phi_1$ . More formally: For  $(G, \mathcal{D}^{sd}) \leftarrow \Gamma_\lambda$ ,  $\phi = (\phi_1, \phi_2, \phi_3) \leftarrow \mathcal{D}^{sd}$ , and  $\mathbf{e} \leftarrow \eta$ , the distributions of  $(\phi_1, G(\phi))$  and  $(\phi_1, \mathbf{e})$  are  $O(\mu)$ -indistinguishable.

We further assume all functions  $G$  in the support of  $\Gamma$  have range contained in  $\text{Support}(\eta)$  and polynomial-stretch  $(\mathbb{Z}_p^{m^{1-\alpha}})^3 \rightarrow \mathbb{Z}_p^m$ . Moreover, for all these functions  $G$ , computing  $G$  and adding a linear function yields a function in the class  $\mathcal{G}$ .

For our construction in [Section 6.3](#), we need an  $\eta$ -noisy FE scheme for a distribution  $\eta$  of the form  $\sum_i p_i \Phi_i$ , where  $p_i$  are integers and  $\Phi_i$  are flawed-smudging distributions. In that case, we can use pseudo flawed-smudging generators  $\text{PFG}_i$  with distributions indistinguishable from  $\Phi_i$  and let  $G$  compute  $\sum_i p_i \text{PFG}_i(\phi_i)$ .

**Construction of NFE.** Our scheme NFE consists of the following algorithms:

- $\text{NFE.Setup}(1^\lambda)$ , on input the security parameter, generates a master secret key  $\text{msk} \leftarrow \text{PHFE.Setup}(1^\lambda)$  for the PHFE scheme, and samples  $(G, \mathcal{D}^{sd}) \leftarrow \Gamma_\lambda$ . It outputs  $\text{nmsk} = (\text{msk}, G, \mathcal{D}^{sd})$ .
- $\text{NFE.Enc}(\text{nmsk}, \mathbf{x})$ , on input  $\text{nmsk} = (\text{msk}, G, \mathcal{D}^{sd})$  and  $x \in \mathbb{Z}_p^n$ , samples  $\phi = (\phi_1, \phi_2, \phi_3) \leftarrow \mathcal{D}^{sd}$ , sets  $X = \phi_1$  and  $Y = (\phi_2, \phi_3 || \mathbf{x})$ , and encrypts  $\text{ct} \leftarrow \text{PHFE.Enc}(\text{msk}, (X, Y))$ . Finally, it outputs the ciphertext  $\text{nct} = \text{ct}$ .
- $\text{NFE.KeyGen}(\text{nmsk}, f)$ , on input  $\text{nmsk} = (\text{msk}, G, \mathcal{D}^{sd})$  and a linear function  $f \in \mathcal{F}$ , does the following: Let  $g$  be the function

$$g(\phi_1, \phi_2, \phi_3 || \mathbf{x}) = f(\mathbf{x}) + G(\phi_1, \phi_2, \phi_3),$$

and generate the key  $\text{nsk}_g \leftarrow \text{PHFE.KeyGen}(\text{msk}, g)$ . Note that  $g \in \mathcal{G}$  by assumption. Output  $\text{nsk}_f = \text{nsk}_g$ .

- $\text{NFE.Dec}(\text{nsk}_f, \text{nct}, \mathcal{R}_1, \dots, \mathcal{R}_m)$  on input  $\text{nsk}_f = \text{nsk}_g$ ,  $\text{nct} = \text{ct}$ , and sets  $\mathcal{R}_1, \dots, \mathcal{R}_m$ , let for  $i \in [m]$ ,

$$\mathcal{R}'_i := \{ \lfloor r_i + e_i \rfloor_p \mid r_i \in \mathcal{R}_i, (e_1, \dots, e_m) \in \text{Support}(\eta) \}.$$

Then compute  $\lfloor \mathbf{y} \rfloor_T = \text{PHFE.Dec}(\text{nsk}_g, \text{ct})$ , and extract each coordinate  $\mathbf{y}_i$  by comparing  $\lfloor \mathbf{y}_i \rfloor_T$  to all  $\lfloor r_i \rfloor_T$  for  $r \in \mathcal{R}'_i$ . Output  $\mathbf{y}$ .

We next prove the correctness, compactness, and security properties of our scheme.

Weak correctness: Let  $\lambda \in \mathbb{N}$ ,  $f \in \mathcal{F}$ ,  $\mathbf{x} \in \mathbb{Z}_p^n$ , and let  $\mathcal{R}_1, \dots, \mathcal{R}_m \subseteq \mathbb{Z}_p$  such that  $\lfloor f(\mathbf{x}) \rfloor_p \in \mathcal{R}_1 \times \dots \times \mathcal{R}_m$ , and all  $\mathcal{R}_i$  are of polynomial size. Further let  $\text{nmsk} \leftarrow \text{NFE.Setup}(1^\lambda)$ ,  $\text{nsk}_f \leftarrow \text{NFE.KeyGen}(\text{nmsk}, f)$ ,  $\text{nct} \leftarrow \text{NFE.Enc}(\text{nmsk}, \mathbf{x})$ , and  $\lfloor \mathbf{y} \rfloor_T = \text{PHFE.Dec}(\text{nsk}_f, \text{nct})$ . Correctness in the exponent of PHFE implies that  $\lfloor \mathbf{y} \rfloor_T = \lfloor g(\phi_1, \phi_2, \phi_3 || \mathbf{x}) \rfloor_T$ , where  $\phi_1, \phi_2, \phi_3$  is the seed encrypted together with  $\mathbf{x}$ . We further have  $\lfloor g_i(\phi_1, \phi_2, \phi_3 || \mathbf{x}) \rfloor_p \in \mathcal{R}'_i$  for all  $i$  since  $\lfloor f_i(\mathbf{x}) \rfloor_p \in \mathcal{R}_i$  and the range of  $G$  is contained in  $\text{Support}(\eta)$ . Hence,  $\text{NFE.Dec}$  can correctly extract

$$\mathbf{y} = \lfloor g(\phi_1, \phi_2, \phi_3 || \mathbf{x}) \rfloor_p = \lfloor f(\mathbf{x}) + G(\phi_1, \phi_2, \phi_3) \rfloor_p.$$

Note that since all  $\mathcal{R}_i$  are of polynomial size and the support of  $\eta$  is polynomially bounded, all  $\mathcal{R}'_i$  used by  $\text{NFE.Dec}$  are of polynomial size, and the extraction can be done efficiently.

Special-purpose sublinear compactness: The algorithm  $\text{NFE.Enc}$  encrypts  $\phi_1$  and  $(\phi_2, \phi_3 || \mathbf{x})$  using the scheme PHFE. We have  $\mathbf{x} \in \mathbb{Z}_p^n$  and  $\phi_i \in \mathbb{Z}_p^{m^{1-\alpha}}$ . Hence, linear efficiency of PHFE implies that the encryption time is  $\text{poly}(\lambda)(n + m^{1-\alpha})$ . Since the ciphertext size output by the encryption can be bounded by the encryption time, the size of its ciphertext is

$$|\text{ct}| \leq \text{poly}(\lambda)(n + m^{1-\alpha}).$$

Note that unlike standard sublinear compactness, which allows the ciphertext size to grow polynomially with the length  $n$  of the input, the ciphertext size of NFE grows only linearly in  $n$ . We refer to this as special-purpose  $(1 - \alpha)$ -sublinear compactness.

*Remark 5.3.* In contrast to the original notion of compactness in [AJ15, BV15] that restricts the encryption time (as we do with linear efficiency of PHFE), we here restrict the ciphertext size. This is because the encryption algorithm  $\text{NFE.Enc}$  samples a seed from some specific distribution, which may take time polynomial in the seed length  $\text{poly}(m^{1-\alpha})$  to sample, and

hence the encryption time may not be compact. We can of course alternatively consider noise generators with seed distributions that are samplable in  $O(m^{1-\alpha})$  time, which would give compact encryption time. But FE with only compact ciphertext size already suffices for constructing IO; see [Section 7.3](#). Therefore, it is better to not constrain the sampling time of seed distributions here.

Simulation security: We finally prove that our scheme is 1-key Sel-Sim-secure.

**Lemma 5.4.** *Assume that PHFE satisfies 1-key  $O(\mu)$ -Sel-PH-Sim-security, and for  $(G, \mathcal{D}^{sd}) \leftarrow \Gamma_\lambda$ ,  $\phi = (\phi_1, \phi_2, \phi_3) \leftarrow \mathcal{D}^{sd}$ , and  $\mathbf{e} \leftarrow \eta$ , the distributions of  $(\phi_1, G(\phi))$  and  $(\phi_1, \mathbf{e})$  are  $O(\mu)$ -indistinguishable. Then, NFE is 1-key  $O(\mu)$ -Sel-Sim-secure.*

*Proof.* Let  $\{f_\lambda\}_{\lambda \in \mathbb{N}}$  be an ensemble of linear functions, let  $\{\mathbf{x}_\lambda^*\}_{\lambda \in \mathbb{N}}$  be an ensemble of inputs, let  $t$  be a polynomial, and let  $\{\mathbf{x}_{i,\lambda}\}_{\lambda \in \mathbb{N}, i \in [t(\lambda)]}$  be an ensemble of sequences of inputs. We prove security via a sequence of hybrids, where the first hybrid is the real distribution from [Definition 5.2](#) and the last one is the ideal distribution. The simulator NSim for the ideal distribution is defined in the last hybrid.

**Hybrid  $H_0$**  is the real distribution. Plugging the algorithms of our scheme into the real distribution of [Definition 5.2](#), we obtain

$$H_0 = \left\{ \begin{array}{l} \text{msk} \leftarrow \text{PHFE.Setup}(1^\lambda) \\ (G, \mathcal{D}^{sd}) \leftarrow \Gamma_\lambda \\ \text{sk}_{g_{f_\lambda, G}} \leftarrow \text{PHFE.KeyGen}(\text{msk}, g_{f_\lambda, G}) \\ \phi^*, \phi_1, \dots, \phi_{t(\lambda)} \leftarrow \mathcal{D}^{sd} \\ \text{ct}^* \leftarrow \text{PHFE.Enc}(\text{msk}, (\phi_1^*, \phi_2^*, \phi_3^* | \mathbf{x}_\lambda^*)) \\ \{\text{ct}_i \leftarrow \text{PHFE.Enc}(\text{msk}, (\phi_{i,1}, \phi_{i,2}, \phi_{i,3} | \mathbf{x}_{i,\lambda}))\}_{i \in [t(\lambda)]} \end{array} : \begin{array}{l} \text{sk}_{g_{f_\lambda, G}}, \text{ct}^*, \\ \{\text{ct}_i\}_{i \in [t(\lambda)]} \end{array} \right\}_{\lambda \in \mathbb{N}},$$

where  $g_{f_\lambda, G}$  is the function  $g_{f_\lambda, G}(\phi_1, \phi_2, \phi_3 | \mathbf{x}) = f_\lambda(\mathbf{x}) + G(\phi_1, \phi_2, \phi_3)$ .

**Hybrid  $H_1$**  generates the secret key  $\text{sk}_{g_{f_\lambda, G}}$  and all ciphertexts using the simulator PSim of the scheme PHFE:

$$H_1 = \left\{ \begin{array}{l} (G, \mathcal{D}^{sd}) \leftarrow \Gamma_\lambda \\ \phi^*, \phi_1, \dots, \phi_{t(\lambda)} \leftarrow \mathcal{D}^{sd} \end{array} : \begin{array}{l} \text{PSim}(g_{f_\lambda, G}, \lfloor g_{f_\lambda, G}(\phi_1^*, \phi_2^*, \phi_3^* | \mathbf{x}_\lambda^*) \rfloor_p, \\ \phi_1^*, \{\mathbf{x}_{i,\lambda}, \phi_i\}_{i \in [t(\lambda)]}) \end{array} \right\}_{\lambda \in \mathbb{N}}.$$

The 1-key  $O(\mu)$ -Sel-PH-Sim-security of PHFE implies that the hybrids  $H_0$  and  $H_1$  are  $O(\mu)$ -indistinguishable.

**Hybrid  $H_2$**  is the same as  $H_1$ , except that we replace  $g_{f_\lambda, G}(\phi_1^*, \phi_2^*, \phi_3^* | \mathbf{x}_\lambda^*) = f_\lambda(\mathbf{x}_\lambda^*) + G(\phi^*)$  in the input of PSim with  $f_\lambda(\mathbf{x}_\lambda^*) + \mathbf{e}$ , where  $\mathbf{e}$  is sampled from  $\eta$ :

$$H_2 = \left\{ \begin{array}{l} (G, \mathcal{D}^{sd}) \leftarrow \Gamma_\lambda \\ \phi^*, \phi_1, \dots, \phi_{t(\lambda)} \leftarrow \mathcal{D}^{sd} \\ \mathbf{e} \leftarrow \eta(\lambda) \end{array} : \begin{array}{l} \text{PSim}(g_{f_\lambda, G}, \lfloor f_\lambda(\mathbf{x}_\lambda^*) + \mathbf{e} \rfloor_p, \\ \phi_1^*, \{\mathbf{x}_{i,\lambda}, \phi_i\}_{i \in [t(\lambda)]}) \end{array} \right\}_{\lambda \in \mathbb{N}}.$$

Since the distributions of  $(\phi_1^*, G(\phi^*))$  and  $(\phi_1^*, \mathbf{e})$  are  $O(\mu)$ -indistinguishable,  $H_1$  and  $H_2$  are  $O(\mu)$ -indistinguishable.

**Hybrid  $H_3$**  is the same as  $H_2$ , but we define  $\text{NSim}$  to produce the outputs. On input  $f_\lambda$ ,  $\mathbf{y}$ , and  $\{\mathbf{x}_{i,\lambda}\}_{i \in [t(\lambda)]}$ , the simulator  $\text{NSim}$  samples  $(G, \mathcal{D}^{sd}) \leftarrow \Gamma_\lambda$  and  $\phi^*, \phi_1, \dots, \phi_{t(\lambda)} \leftarrow \mathcal{D}^{sd}$ . It then defines the function  $g_{f_\lambda, G}(\phi_1, \phi_2, \phi_3 | \mathbf{x}) = f_\lambda(\mathbf{x}) + G(\phi_1, \phi_2, \phi_3)$  and runs the simulator  $\text{PSim}(g_{f_\lambda, G}, \mathbf{y}, \phi_1^*, \{\mathbf{x}_{i,\lambda}, \phi_i\}_{i \in [t(\lambda)]})$ . It outputs the output of  $\text{PSim}$ . The hybrid  $H_3$  is then

$$H_3 = \left\{ \mathbf{e} \leftarrow \eta(\lambda) : \text{NSim}\left(f_\lambda, \lfloor f_\lambda(\mathbf{x}_\lambda^*) + \mathbf{e} \rfloor_p, \{\mathbf{x}_{i,\lambda}\}_{i \in [t(\lambda)]}\right) \right\}_{\lambda \in \mathbb{N}}.$$

The hybrids  $H_2$  and  $H_3$  are identical. Moreover,  $H_3$  corresponds to the ideal distribution in [Definition 5.2](#). We can thus conclude that NFE is 1-key  $O(\mu)$ -Sel-Sim-secure.  $\square$

### 5.3 Instantiating the PHFE and Noise Generator

To construct a NFE scheme for noise distribution  $\eta$ , we need a family of distributions  $\{\Gamma_\lambda\}$  that can sample generator functions and input distributions  $(G, \mathcal{D}^{sd}) \leftarrow \Gamma_\lambda$  such that  $G(\mathcal{D}^{sd})$  is indistinguishable to  $\eta$ . If  $G(\mathbf{x}, \mathbf{y})$  has part of its input public  $\mathbf{x}$  — meaning that the output of  $G(\mathbf{x}, \mathbf{y})$  is indistinguishable to  $\eta$  even when  $\mathbf{x}$  is public — we can compute it using PHFE; otherwise, if all inputs are private, then we can only compute it using FE, see [Section 5.4](#).

Here, we first describe a simple technique to turn  $G(\mathbf{x}, \mathbf{y})$  with private inputs into a function  $H(\mathbf{c}, \mathbf{y}) = G(\mathbf{x}, \mathbf{y})$  whose first input  $\mathbf{c}$  can be made public, if  $G$  satisfies certain properties. Next, we discuss what kind of functions our PHFE schemes from bilinear maps in [Section 4](#) can support.

**Compiling  $G$  with private inputs into  $H$  with partial public input.** The compilation works with functions  $G$  satisfying the following properties.

1.  $G$  is a polynomial over  $\mathbb{Z}_p$ .
2.  $\mathbf{x}$  sampled by  $\mathcal{D}^{sd}$  can be used as noises in LWE samples.
3.  $G$  has at most constant degree  $d = O(1)$  in  $\mathbf{x}$ ; it will be convenient to use a form where  $\mathbf{x} = \mathbf{x}^1, \dots, \mathbf{x}^d$  and  $G(\mathbf{x} = (\mathbf{x}^1, \dots, \mathbf{x}^d), \mathbf{y}, \mathbf{z})$  is multilinear in each  $\mathbf{x}^k$ <sup>10</sup>.
4.  $G$  satisfies the following stronger security property:

$$\{G(\mathbf{x}, \mathbf{y}), \{\mathbf{c}_i\}_{i \in [|\mathbf{x}|]}\} \approx \{\Delta \leftarrow \eta, \{\mathbf{c}_i\}_{i \in [|\mathbf{x}|]}\},$$

where  $(G, \mathcal{D}^{sd}) \leftarrow \Gamma_\lambda$ ,  $(\mathbf{x}, \mathbf{y}) \leftarrow \mathcal{D}^{sd}$ , and  $\mathbf{c}_i = \langle \mathbf{a}_i, \mathbf{s}' \rangle + x_i$  is an LWE sample with random vector  $\mathbf{a}_i \leftarrow \mathbb{Z}_p^m$ , secret  $\mathbf{s}' \leftarrow \mathbb{Z}_p^m$ , and noise  $x_i$ <sup>11</sup>.

The idea is hiding  $\mathbf{x}$  in LWE samples, that is, for every  $x_i$ , generate  $\mathbf{c}_i = (\mathbf{a}_i, \langle \mathbf{a}_i, \mathbf{s}' \rangle + x_i)$ . Since the “decryption” equation is  $x_i = \langle \mathbf{c}_i, \mathbf{s} \rangle$  for  $\mathbf{s} = (-\mathbf{s}' || 1)$ , we can convert  $G$  using  $\mathbf{x}$  into  $H$  using  $\mathbf{c}, \mathbf{s}$  instead.

To see this, consider stratifying  $G$  by monomials of  $\mathbf{x}$  (though the number of monomials may be superpolynomial):

$$G(\mathbf{x} = (\mathbf{x}^1, \dots, \mathbf{x}^d), \mathbf{y}) = \sum_j \left( f_j(\mathbf{y}) \prod_{i \in [d]} x_{j_i}^i \right).$$

<sup>10</sup>If  $G$  has a single input  $\mathbf{x}$ , one can always duplicate same input  $\mathbf{x}$  for  $d$  times to make it multilinear

<sup>11</sup>We can also use different secret keys  $\mathbf{s}$  to encrypt different parts of  $\mathbf{s}$ . For simplicity, we use a single  $\mathbf{s}$ .

By the decryption equation, we have

$$\begin{aligned}
G(\mathbf{x} = (\mathbf{x}^1, \dots, \mathbf{x}^d), \mathbf{y}) &= \sum_j \left( f_j(\mathbf{y}) \prod_{i \in [d]} \langle \mathbf{c}_{j_i}^i, \mathbf{s} \rangle \right) = \sum_j \left( f_j(\mathbf{y}) \left( \sum_{k_1, \dots, k_d \in [m+1]} \prod_i c_{j_i, k_i}^i s_{k_i} \right) \right) \\
&= \sum_{k_1, \dots, k_d} \sum_j \left( f_j(\mathbf{y}) \prod_i c_{j_i, k_i}^i s_{k_i} \right) = \sum_{k_1, \dots, k_d} \prod_i s_{k_i} \left( \sum_j \left( f_j(\mathbf{y}) \prod_i c_{j_i, k_i}^i \right) \right) \\
&= \sum_{k_1, \dots, k_d} \left( \prod_i s_{k_i} \right) G(\mathbf{c}_{\star, k_1}^1, \dots, \mathbf{c}_{\star, k_d}^d, \mathbf{y}),
\end{aligned}$$

where  $\mathbf{c}_{\star, k_i}^i$  denotes the concatenation of  $\{c_{j, k_i}^i\}_{j \in N}$ . Now let  $\mathbf{s}^{(d)}$  denote the vector obtained by tensoring  $\mathbf{s}$  for  $d$  times. Assume without loss of generality that  $y$  contains 1. We define  $H$  to be

$$H(\mathbf{c}, \mathbf{y}' = \mathbf{y} \otimes \mathbf{s}^{(d)}) := \sum_{k_1, \dots, k_d \in [m+1]} \left( \prod_i s_{k_i} \right) G(\mathbf{c}_{\star, k_i}^1, \dots, \mathbf{c}_{\star, k_d}^d, \mathbf{y}).$$

We observe that  $h$  also has degree  $d$  in  $\mathbf{c}$  and its degree in  $\mathbf{y}' = \mathbf{y} \otimes \mathbf{s}^{(d)}$  is the same as the degree of  $g$  in  $\mathbf{y}$ . The length of the private input  $|\mathbf{y}'|$  is  $|\mathbf{y}|(m+1)^d$ , and the complexity of  $H$  is a multiplicative  $(m+1)^d$  factor higher than that of  $G$ . Since  $d$  is a constant, the blow up is polynomial. We denote by  $\{\mathcal{H}_\lambda\}$  the distributions that sample  $H$  and its input distribution  $\mathcal{D}_H$ .

**Degree 3 PHFE supports degree 3 generators.** Section 4.1 presents a PHFE scheme PHFE for degree 3 multilinear polynomials  $g(\mathbf{x}, \mathbf{y}, \mathbf{z})$ , where  $\mathbf{x}$  is public, based on the SXDH assumption from bilinear maps. Suppose we have the following.

- **A degree 3 multilinear PFG**  $\{\text{PFG}_\lambda\}$  with the following strong indistinguishability: Its outputs are indistinguishable to a family of  $(K, \mu)$ -flawed-smudging distributions  $\{\mathcal{X}_\lambda\}$ . The indistinguishability holds even at the presence of LWE samples with  $\mathbf{x}$  as noises.

$$\{g(\mathbf{x}, \mathbf{y}, \mathbf{z}), \{\mathbf{c}_i\}_{i \in [|\mathbf{x}|]}\} \approx \{\Delta \leftarrow \mathcal{X}, \{\mathbf{c}_i\}_{i \in [|\mathbf{x}|]}\}, \text{ where } \mathbf{c}_i = (\mathbf{a}_i, \langle \mathbf{a}_i, \mathbf{s}' \rangle + x_i).$$

Applying the above transformation gives

- **A degree 3 multilinear PFG with one public input**  $\{\mathcal{H}_\lambda\}$  satisfying that its outputs are indistinguishable to  $\mathcal{X}$  even when  $\mathbf{c}$  is public

$$\{h(\mathbf{c}, (\mathbf{y}' = \mathbf{y} \otimes \mathbf{s}), \mathbf{z}), \mathbf{c}\} \approx \{\Delta \leftarrow \mathcal{X}, \mathbf{c}\}.$$

Furthermore, if  $\Gamma$  has input lengths  $N, N/(m+1), N$ ,  $\mathcal{H}$  has input lengths  $N, N, N$ , where  $m = |\mathbf{s}|$  is the length of the LWE secret. The choice of  $m$  may affect security; it can be for instance  $\text{poly}(\lambda)$  or  $N^{0.5}$ .

Therefore,  $\mathcal{H}$  can be computed by PHFE in Section 4.1 for constructing NFE for flawed-smudging distributions.

*Remark 5.5 (Relation to the degree 3 candidate  $\Delta\text{RG}$  in [AJKS18]).* We emphasize that the transformed degree 3 PFG  $\mathcal{H}$  has form identical to the degree 3  $\Delta\text{RG}$  proposed in [AJKS18]. Our generic transformation above is inspired by their  $\Delta\text{RG}$  candidate, and  $\mathcal{H}$  is a direct adaptation of their  $\Delta\text{RG}$  to have the flawed-smudging property ( $\Delta\text{RG}$  has a different security notion).

Below, we further extend their candidate to consider PFGs with higher constant degree in the public input. The reason that we can support them is because we managed to construct PHFE able to support them from bilinear maps.

**Constant degree PHFE supports constant-degree canonical generators.** In Section 4.2, building upon the degree 3 scheme, we constructed PHFE scheme  $\text{PHFE}^{\text{Dep}}$  for the special class of polynomials  $g \in \mathcal{G}^{N,S,\text{Dep}}$ . Recall that they have canonical form  $g(\mathbf{x}, \mathbf{y}, \mathbf{z}) = \langle\langle \alpha(\mathbf{y}, \mathbf{z}), f(\ell(\mathbf{x})) \rangle\rangle + \beta(\mathbf{y}, \mathbf{z})$ , and satisfy the constraints that  $\alpha, \beta, \ell$  are multilinear in  $\mathbf{x}, \mathbf{y}, \mathbf{z}$ ,  $f$  is a logarithmic depth  $\text{Dep} = O(\log(\lambda))$  formula, and  $\max(\text{width}(f), |g|) \leq N^2$ . Since our transformation above only works for polynomials with constant degree  $d = O(1)$  in  $\mathbf{x}$ , we restrict ourselves to constant depth  $\text{Dep} = O(1)$ . Suppose we have

- A degree- $d$  PFG in  $\mathcal{G}^{N,S,\text{Dep}}$ ,  $\{\text{PFG}_\lambda\}$ , with the same strong indistinguishability as above: For some  $(K, \mu)$ -flawed-smudging distributions  $\{\mathcal{X}_\lambda\}$ ,

$$\{g(\mathbf{x}, \mathbf{y}, \mathbf{z}), \{\mathbf{c}_i\}_{i \in [|\mathbf{x}|]}\} \approx \{\Delta \leftarrow \mathcal{X}, \{\mathbf{c}_i\}_{i \in [|\mathbf{x}|]}\}, \text{ where } \mathbf{c}_i = (\mathbf{a}_i, \langle \mathbf{a}_i, \mathbf{s}' \rangle + x_i).$$

A careful examination shows that applying the above transformation preserves the canonical form, as, roughly speaking,  $h$  essentially makes  $(m+1)^d$  calls of  $g$  and sums their outputs up with monomials of  $\mathbf{s}$  as coefficient. This gives,

- A degree- $d$  PFG in  $\mathcal{H}^{N,S',\text{Dep}}$  with public input,  $\{\mathcal{H}_\lambda\}$ , satisfying that

$$\{h(\mathbf{c}, (\mathbf{y}' = \mathbf{y} \otimes \mathbf{s}^{(d)}), \mathbf{z}), \mathbf{c}\} \approx \{\Delta \leftarrow \mathcal{X}, \mathbf{c}\}.$$

Furthermore, if  $g$  in  $\mathcal{G}$  has input lengths  $N, N/(m+1)^d, N$ ,  $\mathcal{H}$  has input lengths  $N, N, N$ , where  $m = |\mathbf{s}|$ . If  $g$  has that  $\text{width}(f) \leq N^2/(m+1)^d$ , then  $h$  has the width of its corresponding  $f_h$  bounded by  $N^2$ . Again, the choice of  $m$  may affect security; one can consider for instance  $\text{poly}(\lambda)$  or  $N^\varepsilon$  for  $\varepsilon < 1/2d$ .

Therefore,  $\mathcal{H}$  can be computed by  $\text{PHFE}^{\text{Dep}}$  from Section 4.2 for constructing NFE for flawed-smudging distributions.

## 5.4 Construction from FE and Noise Generator

Our construction in Section 5.2 uses a special type of noise generators that remain secure when a part of the seed is public. We can replace that by regular noise generators (which are only secure if all of their seeds remain hidden) if we use a regular FE scheme instead of PHFE. We here consider the construction from an FE scheme for polynomials of degree  $d$  over  $\mathbb{Z}_p$  and noise generators with output distributions indistinguishable from  $\eta$  that can be computed by degree- $d$  polynomials over  $\mathbb{Z}_p$ .

More concretely, we assume as building blocks such noise generators  $\{\Gamma_\lambda\}_{\lambda \in \mathbb{N}}$  with polynomial-stretch  $\mathbb{Z}^{m^{1-\alpha}} \rightarrow \mathbb{Z}^m$ , and secret key FE schemes  $\text{FE}^{n,m}$  for computing polynomials of degree  $d$  over  $\mathbb{Z}_p$  mapping  $\mathbb{Z}_p^{n+m^{1-\alpha}} \rightarrow \mathbb{Z}_p^m$ , with correctness in the exponent, linear efficiency, and 1-key  $O(\mu)$ -Sel-Sim-security. Such FE schemes are constructed from degree  $d$  multilinear maps by Lin [Lin17]. As discussed in Section 4.1, these schemes satisfy our requirements, in particular simulation security.

From these building blocks, we construct an  $\eta$ -noisy secret-key FE scheme NFE for linear functions  $\mathbb{Z}_p^n \rightarrow \mathbb{Z}_p^m$ . The construction is almost identical to the one in Section 5.2. We here use the scheme FE instead of PHFE and since the noise generator here has no public and private parts, we simply encrypt  $(\mathbf{x}, \phi)$  instead of  $(\phi_1, \phi_2, \phi_3 || \mathbf{x})$ , and to generate a key for a function  $f$ ,  $\text{NFE.KeyGen}$  generates an FE key for the function  $g(\mathbf{x}, \phi) = f(\mathbf{x}) + G(\phi)$ . Weak correctness and special-purpose sublinear compactness follow as in Section 5.2. Furthermore, the scheme is 1-key  $O(\mu)$ -Sel-Sim-secure, as shown in the following lemma.

**Lemma 5.6.** *Assume that FE satisfies 1-key  $O(\mu)$ -Sel-Sim-security, and the outputs of  $\Gamma$  are  $O(\mu)$ -indistinguishable from  $\eta$ . Then, NFE also is 1-key  $O(\mu)$ -Sel-Sim-secure.*

*Proof sketch.* The proof is almost identical to the proof of Lemma 5.4. We again start with hybrid  $H_0$  as the real distribution. Since we here use an FE scheme with 1-key  $O(\mu)$ -Sel-Sim-security, the simulator in hybrid  $H_1$  does not receive the public part of the seed  $\phi_1^*$ , i.e.,  $H_1$  outputs

$$\text{Sim}\left(g_{f_\lambda, G}, \left[g_{f_\lambda, G}(\mathbf{x}_\lambda^*, \phi^*)\right]_p, \{\mathbf{x}_{i, \lambda}, \phi_i\}_{i \in [t(\lambda)]}\right).$$

In  $H_2$ , we then use the properties of  $\Gamma$  to replace  $g_{f_\lambda, G}(\mathbf{x}_\lambda^*, \phi^*) = f_\lambda(\mathbf{x}_\lambda^*) + G(\phi)$  in the input of Sim with  $f_\lambda(\mathbf{x}_\lambda^*) + \mathbf{e}$ , where  $\mathbf{e}$  is sampled from  $\eta$ . Finally, hybrid  $H_3$ , which corresponds to the ideal distribution in Definition 5.2, defines NSim identically as in the proof of Lemma 5.4 except that it does not need to sample  $\phi^*$ , since it is not needed by Sim.  $\square$

## 6 Functional Encryption for Constant Degree Polynomials

In this section, for any degree  $D \in \mathbb{Z}$  and any polynomially large modulus  $p_D$ , we construct special-purpose FE schemes for computing degree  $D$  polynomials over  $\mathbb{Z}_{p_D}$ . We will show that our FE schemes satisfies *special-purpose* sub-linear compactness and (1-key) simulation-security. Our construction makes use of the simple secret-key Homomorphic Encryption (HE) scheme supporting evaluation of *constant-degree* polynomials proposed by Brakerski and Vaikuntanathan (BV) [BV11], referred to as the *simple-BV* scheme. We next review this scheme and observe that it has many amenable properties that are instrumental for our construction of FE.

### 6.1 Review of the Simple-BV HE Scheme

Brakerski and Vaikuntanathan showed that the simple secret-key encryption scheme based on the hardness of LWE — a ciphertext of  $x \in \mathbb{Z}_p$  is of the form  $\mathbf{a}, \langle \mathbf{a}, \mathbf{s} \rangle + pe + x$  over  $\mathbb{Z}_q$  — supports homomorphic evaluation of constant-degree polynomials. (Without relinearization and bootstrapping, the ciphertext-size grows exponentially with the depth of the computation, but it suffices for evaluating constant-degree polynomials.) Furthermore, as shown in [GKPV10, AKPW13], when the secret  $\mathbf{s}$  is from  $\{-1, 0, 1\}$ , the hardness of LWE holds even if the secret  $\mathbf{s}$  is only entropic instead of uniformly random. This means the semantic security of the scheme holds as long as  $\mathbf{s}$  has sufficient entropy. Below we describe the scheme, and highlight the properties that will be useful for our construction of FE schemes.

The simple-BV HE scheme HE consists of the following algorithms. They are parameterized by public parameters  $\mathbf{pp} = (p, q, \chi)$ , where  $p < q$  are coprime integers and  $\chi$  is an  $\Upsilon$ -bounded distribution over  $\mathbb{Z}_q$ . We set these parameters in Remark 6.5. The public parameters are specified as a subscript of the algorithms, and omitted when they are clear in the context. We emphasize that all computations are by default over the integers; all modular operations are explicitly denoted by  $[x]_p = x \bmod p$ . In the description below, we consider only functions with a single output element; functions with multiple output elements are handled component-wise as described in Remark 6.2.

- $\text{HE.KeyGen}_{\mathbf{pp}}(1^n)$ , on input a security parameter  $1^n$ , outputs a random vector  $\mathbf{s} \leftarrow \{-1, 0, 1\}^n$ .

- $\text{HE.Enc}_{\text{pp}}(\mathbf{s}, x)$ , on input a message  $x \in \mathbb{Z}_p$ , samples a vector  $\mathbf{a} \leftarrow \mathbb{Z}_q^n$  and noise  $e \leftarrow \chi$ , and outputs

$$\text{hct} = \left( \mathbf{a}, c = \lfloor \langle \mathbf{a}, \mathbf{s} \rangle + pe + x \rfloor_q \right) .$$

- $\text{HE.Eval}_{\text{pp}}(f, \text{hct}_1, \dots, \text{hct}_t)$  on input a degree  $d$  polynomial  $f$  mapping  $t$  input elements to a single output element, and  $t$  ciphertexts, expands  $f$  as a sum of monomials<sup>12</sup>

$$f(\mathbf{x}) = \sum_j \text{mnl}_j(\mathbf{x}) , \text{ where } \text{mnl}_j(\mathbf{x}) = \prod_{\gamma \in [d]} x_{j_\gamma} .$$

It then outputs the ciphertext

$$\text{hct}^f = \sum_j \text{hct}^{\text{mnl}_j} , \text{ where } \text{hct}^{\text{mnl}_j} = \otimes_{\gamma \in [d]} \text{hct}_{j_\gamma} ,$$

where  $\otimes$  stands for the vector tensor product.

- $\text{HE.Dec}_{\text{pp}}(\mathbf{s}, \text{hct})$ : The decryption equation for a freshly generated ciphertext  $\text{hct}$  encrypting  $x$  is

$$\lfloor \langle \text{hct}, \mathbf{s}' \rangle \rfloor_q = \lfloor x + pe \rfloor_q , \quad \text{where } \mathbf{s}' = (-\mathbf{s} || 1) .$$

Thus, the decryption equation for a ciphertext  $\text{hct} = \text{hct}^{\text{mnl}_j}$  derived from homomorphically evaluating a degree  $d$  monomial  $\text{mnl}_j(\mathbf{x})$  over ciphertexts  $\text{hct}_1, \dots, \text{hct}_t$  encrypting  $\mathbf{x}$  with noises  $\mathbf{e}$  is

$$\begin{aligned} \lfloor \langle \text{hct}^{\text{mnl}_j}, (\mathbf{s}'^{\otimes d}) \rangle \rfloor_q &= \lfloor \text{mnl}_j(\mathbf{x} + p\mathbf{e}) \rfloor_q \\ &= \lfloor \text{mnl}_j(\mathbf{x}) + pe^{\text{mnl}_j} \rfloor_q = \text{mnl}_j(\mathbf{x}) + pe^{\text{mnl}_j} , \end{aligned} \quad (8)$$

where  $\mathbf{s}'^{\otimes d} = \otimes_{\gamma \in [d]} \mathbf{s}'$ ,  $e^{\text{mnl}_j}(\mathbf{x}, \mathbf{e}) = \text{mnl}_j(\mathbf{x} + p\mathbf{e}) - \text{mnl}_j(\mathbf{x})$  is a polynomial over  $\mathbf{x}$  and  $\mathbf{e}$ , and the last equality holds if  $q$  is sufficiently large comparing to  $\text{mnl}_j(\mathbf{x}) + pe^{\text{mnl}_j}$ . In addition, since  $\text{mnl}_j(\mathbf{x}) = \prod_{\gamma \in \Gamma(\text{mnl}_j)} x_\gamma$  depends on only  $d$  variables  $x_\gamma$  for  $\gamma \in \Gamma(\text{mnl}_j)$ ,  $e^{\text{mnl}_j}$  depends only on  $x_\gamma$  and  $e_\gamma$  for  $\gamma \in \Gamma(\text{mnl}_j)$ . That is,

$$e^{\text{mnl}_j} \left( \{ \langle \mathbf{a}_\gamma, \mathbf{s} \rangle, x_\gamma, e_\gamma \}_{\gamma \in \Gamma(\text{mnl}_j)} \right) = \text{mnl}_j \left( \{ \mathbf{x}_\gamma + p\mathbf{e}_\gamma \}_{\gamma \in \Gamma(\text{mnl}_j)} \right) - \text{mnl}_j \left( \{ \mathbf{x}_\gamma \}_{\gamma \in \Gamma(\text{mnl}_j)} \right) .$$

(Though  $\langle \mathbf{a}_\gamma, \mathbf{s} \rangle$  is redundant for the computation of  $e^{\text{mnl}_j}$ , it will be convenient for later to include them in the inputs for computing  $e^{\text{mnl}_j}$ .)

Then, the decryption equation for a ciphertext  $\text{hct} = \text{hct}^f$  derived from homomorphically evaluating a degree  $d$  function  $f$  over ciphertexts  $\text{hct}_1, \dots, \text{hct}_t$  encrypting  $\mathbf{x}$  with noises  $\mathbf{e}$  is

$$\lfloor \langle \text{hct}^f, (\mathbf{s}'^{\otimes d}) \rangle \rfloor_q = \lfloor f(\mathbf{x}) + p \sum_j e^{\text{mnl}_j} \rfloor_q = f(\mathbf{x}) + p \sum_j e^{\text{mnl}_j} = f(\mathbf{x}) + pe^{f^d} , \quad (9)$$

where again the second last equality holds if  $q$  is sufficiently large, exceeding  $f(\mathbf{x}) + p \sum_j e^{\text{mnl}_j}$ , and

$$e^{f^d} := \sum_j e^{\text{mnl}_j} . \quad (10)$$

<sup>12</sup>Assume w.l.o.g. that  $f$  is homogeneous, that is, all monomials have exactly degree  $d$ .

From the right hand side of [equation \(9\)](#), the decryption algorithm computes the output of  $f(\mathbf{x})$  over  $\mathbb{Z}_p$  as

$$y = \lfloor f(\mathbf{x}) \rfloor_p = \lfloor f(\mathbf{x}) + pe^{f^d} \rfloor_p .$$

For correctness, we need to ensure that  $f(\mathbf{x}) + pe^{f^d} < q$ .

Below, we point out several special properties of the simple-BV scheme.

*Property 1 — Decryption over the Integers.* The LHSs of above decryption equations perform modulo  $q$ . For our construction of FE later, we need decryption equations over the integers. From [equation \(8\)](#), we derive the following decryption equation over integers for  $\text{mnl}_j$ :

$$\langle \text{hct}^{\text{mnl}_j}, (\mathbf{s}'^{\otimes d}) \rangle = \text{mnl}_j(\mathbf{x}) + pe^{\text{mnl}_j} + qo^{\text{mnl}_j} ,$$

where  $o^{\text{mnl}_j} = \lfloor \langle \text{hct}^{\text{mnl}_j}, (\mathbf{s}'^{\otimes d}) \rangle / q \rfloor$ . For a monomial  $\text{mnl}_j(\mathbf{x}) = \prod_{\gamma \in \Gamma(\text{mnl}_j)} x_\gamma$  depending only on  $d$  variables  $x_\gamma$  for  $\gamma \in \Gamma(\text{mnl}_j)$ ,  $\text{hct}^{\text{mnl}_j}$  depends only on ciphertexts  $\text{hct}_\gamma$  for  $\gamma \in \Gamma(\text{mnl}_j)$  (among the ciphertexts  $\text{hct}_1, \dots, \text{hct}_t$  encrypting  $\mathbf{x}$  with noises  $\mathbf{e}$ ). Since each  $\text{hct}_\gamma$  can be computed from  $\langle \mathbf{a}_\gamma, \mathbf{s} \rangle, x_\gamma, e_\gamma$ , we have that  $o^{\text{mnl}_j}$  is a function on these variables. More precisely,

$$o^{\text{mnl}_j}(\{ \langle \mathbf{a}_\gamma, \mathbf{s} \rangle, x_\gamma, e_\gamma \}_{\gamma \in \Gamma(\text{mnl}_j)}) = \left\lfloor \frac{\prod_{\gamma} (c_\gamma - \langle \mathbf{a}_\gamma, \mathbf{s} \rangle)}{q} \right\rfloor ,$$

where  $c_\gamma = \lfloor \langle \mathbf{a}_\gamma, \mathbf{s} \rangle + x_\gamma + pe_\gamma \rfloor_q$ .

From the above, and the decryption equation [\(9\)](#) for a polynomial  $f$  over  $\mathbb{Z}_q$ , we obtain the decryption equation for  $f$  over integers:

$$\langle \text{hct}^f, (\mathbf{s}'^{\otimes d}) \rangle = f(\mathbf{x}) + p \sum_j e^{\text{mnl}_j} + q \sum_j o^{\text{mnl}_j} .$$

We refer to  $e^{\text{mnl}_j}$  and  $o^{\text{mnl}_j}$  as the decryption noises over integers. We emphasize that they are different from the noises derived when decryption is performed over  $\mathbb{Z}_q$ .

We summarize the above decryption equation over integers in the following claim and note about the additive and local structure of the decryption noises over integers.

**Claim 6.1** (Decryption equation over integers). *Let  $p, q, \chi$  be public parameters described above, where  $\chi$  is  $\Upsilon$ -bounded. The decryption equation over  $\mathbb{Z}$  is: For any honestly generated ciphertexts  $\text{hct}_1, \dots, \text{hct}_t$  encrypting messages  $\mathbf{x} = (x_1, \dots, x_t)$  with secret  $\mathbf{s}$  and noises  $\mathbf{e} = (e_1, \dots, e_t)$ , any degree  $d$  polynomial  $f(\mathbf{x}) = \sum_j \text{mnl}_j(\mathbf{x})$ , where  $\text{mnl}_j(\mathbf{x}) = \prod_{\gamma \in \Gamma(\text{mnl}_j)} x_\gamma$ , and  $\text{hct}^f = \text{HE.Eval}_{\text{pp}}(f, \text{hct}_1, \dots, \text{hct}_t)$ ,*

$$\langle \text{hct}^f, (\mathbf{s}'^{\otimes d}) \rangle = f(\mathbf{x}) + pe^f + qo^f , \quad \text{where } e^f := \sum_j e^{\text{mnl}_j}, \text{ and } o^f := \sum_j o^{\text{mnl}_j} .$$

- Every  $e^{\text{mnl}_j} = e^{\text{mnl}_j}(\{ \langle \mathbf{a}_\gamma, \mathbf{s} \rangle, x_\gamma, e_\gamma \}_{\gamma \in \Gamma(\text{mnl}_j)})$  depends only on variables for locations  $\gamma \in \Gamma(\text{mnl}_j)$ , and is  $\text{poly}(p, \Upsilon)$ -bounded.
- Every  $o^{\text{mnl}_j} = o^{\text{mnl}_j}(\{ \langle \mathbf{a}_\gamma, \mathbf{s} \rangle, x_\gamma, e_\gamma \}_{\gamma \in \Gamma(\text{mnl}_j)})$  depends only on variables for location  $\gamma \in \Gamma(\text{mnl}_j)$ , and is  $\text{poly}(q)$ -bounded if  $n < q$ .

*Property 2 — Public and Private Evaluation-Decryption Operations over Integers.* We note that the evaluation followed by decryption operations,  $\text{HE.Eval}(f, \text{hct}_1, \dots, \text{hct}_t)$  and  $\langle \text{hct}^f, (\mathbf{s}'^{\otimes d}) \rangle$ , over the integers, can be decomposed into a public and a private operation, where the public operation depends only on the ciphertexts  $\{\text{hct}_i = (\mathbf{a}_i, c_i)\}$  and private operation additionally depends on the secret  $\mathbf{s}$ . We treat the public vectors  $\mathbf{A}^T = (\mathbf{a}_1 | \mathbf{a}_2 | \dots | \mathbf{a}_t) \in \mathbb{Z}_q^{n \times t}$  as coefficients of these operations instead of variables, and obtain

$$\langle \text{hct}^f, (\mathbf{s}'^{\otimes d}) \rangle = \text{Pub}^{f, \mathbf{A}}(\mathbf{c}) + \text{Priv}^{f, \mathbf{A}}(\mathbf{c}, \mathbf{s}) .$$

It is easy to observe that

$$\langle \text{hct}^f, (\mathbf{s}'^{\otimes d}) \rangle = f(\mathbf{X}) , \text{ where } \mathbf{X} = \mathbf{c} - \mathbf{A}\mathbf{s} .$$

Therefore, when  $\mathbf{A}$  is treated as coefficients, both  $\text{Pub}^{f, \mathbf{A}}$  and  $\text{Priv}^{f, \mathbf{A}}$  have degree  $d$ , and since all monomials that are independent of  $\mathbf{s}$  are included in  $\text{Pub}^{f, \mathbf{A}}$ , every monomial in  $\text{Priv}^{f, \mathbf{A}}$  has at least degree 1 in  $\mathbf{s}$ , and thus has at most degree  $d - 1$  in  $\mathbf{c}$ . Hence, there is a degree  $d - 1$  polynomial  $\text{Priv}^{f, \mathbf{A}}$  that on input  $(\mathbf{c}, 1) \otimes (\mathbf{s}, 1)^{\otimes d}$  (where all multiplication with  $\mathbf{s}$  has been precomputed) computes  $\text{Priv}^{f, \mathbf{A}}(\mathbf{c}, \mathbf{s})$ . Thus,

$$\langle \text{hct}^f, (\mathbf{s}'^{\otimes d}) \rangle = \text{Pub}^{f, \mathbf{A}}(\mathbf{c}) + \text{Priv}^{f, \mathbf{A}}\left((\mathbf{c}, 1) \otimes (\mathbf{s}, 1)^{\otimes d}\right) .$$

We refer to  $\text{Pub}^{f, \mathbf{A}}$  as the public evaluation-decryption function, and  $\text{Priv}^{f, \mathbf{A}}$  as the private one. Observe that their sizes are both  $\text{poly}(n, |f|)$ . Furthermore, since the LHS is additive w.r.t. different monomials in  $f$ , each term in the RHS is also additive:

$$\sum_j \langle \text{hct}^{\text{mnl}_j}, (\mathbf{s}'^{\otimes d}) \rangle = \left( \sum_j \text{Pub}^{\text{mnl}_j, \mathbf{A}}(\mathbf{c}) \right) + \left( \sum_j \text{Priv}^{\text{mnl}_j, \mathbf{A}}\left((\mathbf{c}, 1) \otimes (\mathbf{s}, 1)^{\otimes d}\right) \right) ,$$

and every  $\text{Pub}^{\text{mnl}_j, \mathbf{A}}$  and  $\text{Priv}^{\text{mnl}_j, \mathbf{A}}$  has size  $\text{poly}(n)$ .

*Remark 6.2* (A note on evaluating function with multiple output elements.). Homomorphic evaluation of a function  $f : \mathbb{Z}_p^t \rightarrow \mathbb{Z}_p^M$  with multiple output elements is done component-wise:

$$\text{HE.Eval}(f, \text{hct}_1, \dots, \text{hct}_t) := \text{hct}^f = \left\{ \text{hct}^{f_i} = \text{HE.Eval}(f_i, \text{hct}_1, \dots, \text{hct}_t) \right\}_{i \in [M]} .$$

For each component, the evaluation-decryption operation can be decomposed into

$$\langle \text{hct}^{f_i}, (\mathbf{s}'^{\otimes d}) \rangle = \text{Pub}^{f_i, \mathbf{A}}(\mathbf{c}) + \text{Priv}^{f_i, \mathbf{A}}\left((\mathbf{c}, 1) \otimes (\mathbf{s}, 1)^{\otimes d}\right) .$$

Naturally, we define  $\text{Pub}^{f, \mathbf{A}} = \{\text{Pub}^{f_i, \mathbf{A}}\}_{i \in [M]}$ ,  $\text{Priv}^{f, \mathbf{A}} = \{\text{Priv}^{f_i, \mathbf{A}}\}_{i \in [M]}$ , and write

$$\left\{ \langle \text{hct}^{f_i}, (\mathbf{s}'^{\otimes d}) \rangle \right\} = \text{Pub}^{f, \mathbf{A}}(\mathbf{c}) + \text{Priv}^{f, \mathbf{A}}\left((\mathbf{c}, 1) \otimes (\mathbf{s}, 1)^{\otimes d}\right) .$$

*Property 3 — Robustness under entropic secrets.* Semantic security of HE follows directly from the hardness of  $\overline{\text{LWE}}$  with binary secrets. Therefore, by the robustness of  $\overline{\text{LWE}}$  [GKPV10, AKPW13], semantic security holds as long as the binary  $\overline{\text{LWE}}$  secret has sufficient entropy. More precisely, we have the following lemma:

**Lemma 6.3** (Robustness of HE). *Let  $q, p, \chi$  be as described above. By the  $\mu$ -indistinguishability of  $\text{LWE}_{n,t,q,\chi}^{\text{WL}(1,k)}$  (see [Definition 2.5](#)), it holds that for all efficiently samplable correlated random variables  $(\mathbf{s}, \text{aux})$ , where the support of  $\mathbf{s}$  is  $\{-1, 0, 1\}^n$  and  $H_\infty(\mathbf{s} \mid \text{aux}) \geq k$ , the following distributions are  $\mu$ -indistinguishable*

$$(\text{aux}, \{\text{hct} \leftarrow \text{HE.Enc}(\mathbf{s}, x_i)\}_{i \in [t]}) \quad (\text{aux}, \{\text{hct} \leftarrow \text{HE.Enc}(\mathbf{s}, 0)\}_{i \in [t]}) ,$$

where  $\{x_i\}_{i \in [t]}$  are arbitrary messages in  $\mathbb{Z}_p$  and the encryption randomness is independent of  $(\mathbf{s}, \text{aux})$ .

It was shown first in [\[GKPV10\]](#) that the weak and leaky LWE assumption is implied by standard LWE. However, their result requires super-polynomial modulus ( $q$  here) and modulus-to-noise ratio, which is insufficient for our purpose. Fortunately, a later work by [\[AKPW13\]](#) improved the result to work with polynomial modulus and modulus-to-noise ratio. We recall their theorem.

**Theorem 6.4** ([\[AKPW13, Theorem B.5\]](#)). *Let  $k, \ell, t, n, \beta, \gamma, \sigma$ , and  $q$  be integers, and let  $\Psi$  be a distribution (all parameterized by  $\lambda$ ) such that  $\Pr_{x \leftarrow \Psi}[|x| \geq \beta] \leq \text{negl}(\lambda)$  and  $\sigma \geq \beta \gamma n t$ . Further let  $\chi_\sigma$  be either the discrete Gaussian distribution with standard deviation  $\sigma$ , or the uniform distribution over  $[-\sigma, \sigma] \cap \mathbb{Z}$ . Assuming that the  $\text{LWE}_{\ell,t,q,\Psi}$ -assumption holds, the weak and leaky  $\text{LWE}_{n,t,q,\chi_\sigma}^{\text{WL}(\gamma,k)}$ -assumption holds if  $k \geq (\ell + \Omega(\lambda)) \log(q)$ , with polynomial security loss.*

*Remark 6.5.* [Setting Parameters for Correctness and Robustness] We now describe how to set the public parameters  $p, q, \chi$  to satisfy both correctness and robustness. For correctness, we need to ensure that  $f(\mathbf{x}) + p \sum_j e^{\text{mnl}_j} < q$ . Let  $s$  be an upper bound on the number of monomials contained in  $f$ , then  $|f(\mathbf{x})| = \text{poly}(s, p)$ , and  $\sum_j e^{\text{mnl}_j}$  is  $s \cdot \text{poly}(p, \Upsilon)$  bounded. Therefore,  $q$  needs to be a sufficiently large polynomial  $\text{poly}(s, p, \Upsilon)$ .

For robustness, by [Theorem 6.4](#), the parameter  $\sigma$  of the noise distribution  $\chi_\sigma$  needs to be a polynomial in  $n, \gamma, \beta$ , and the number  $t$  of LWE samples. By setting  $\gamma = 1$ ,  $\beta = \text{poly}(n)$ , we obtain  $\sigma = \text{poly}(n, t)$  and the noise distribution is  $\Upsilon = \text{poly}(n, t)$  bounded.

Combining the above, it suffices to have  $q = \text{poly}(s, p, \lambda, t)$ .

## 6.2 HE Schemes with Linear Private Evaluation-Decryption Functions

The evaluation-decryption of simple-BV for a degree  $d$  polynomial  $f$  can be decomposed into a degree  $d$  public function  $\text{Pub}^{f,\mathbf{A}}$  and a degree  $d - 1$  private function  $\text{Priv}^{f,\mathbf{A}}$  (see property 2 above). By a simple iterative construction, we obtain, for any integer  $D$ , a HE scheme DHE for evaluating degree  $D$  polynomials that has a *linear* private evaluation-decryption function. In addition, this scheme inherits the nice properties of simple-BV. The homomorphic evaluation of this scheme is however restricted to operating on a single ciphertext; correspondingly, the encryption algorithm directly encrypts vectors  $\mathbf{x}$  instead of scalars.

The scheme DHE consists of the following algorithms. All algorithms are parameterized by public parameters  $\text{pp} = (p_D, q_D, \dots, p_2, q_2, p_1, \chi)$ , where  $p_D < q_D < \dots < p_1$  is a sequence of moduli such that for every  $2 \leq d \leq D$ ,  $p_d, q_d$  are coprime, and  $\chi$  is an  $\Upsilon$ -bounded distribution over  $\mathbb{Z}_q$ . These parameters must satisfy the conditions described in [Remark 6.6](#). The description below handles only functions  $f$  with a single output element; functions with multiple output elements can be handled component-wise as described in [Remark 6.2](#). In addition, we directly describe the evaluation-decryption operation over the integers.

- $\text{DHE.KeyGen}_{\text{pp}}(1^n)$  on input a security parameter  $1^n$ , outputs  $D - 1$  random vector  $\mathbf{s}^{\leq D} = \{\mathbf{s}^d \leftarrow \{-1, 0, 1\}^n\}_{2 \leq d \leq D}$ .
- $\text{DHE.Enc}_{\text{pp}}(\mathbf{s}^{\leq D}, \mathbf{x})$ , on input a vector  $\mathbf{x} \in \mathbb{Z}_{p_D}^t$ , sets  $\mathbf{x}^D = \mathbf{x}$  and performs the following two steps for  $D - 1$  iterations: In iteration  $d = D$  to  $2$ ,
  1. Encrypt  $\text{hct}^d \leftarrow \text{HE.Enc}_{(p_d, q_d, \chi)}(\mathbf{s}^d, \mathbf{x}^d)$ .  
Let  $t^d$  be the number of elements in  $\mathbf{x}^d$ . The ciphertexts  $\text{hct}^d$  consists of  $(\mathbf{A}^d)^T = (\mathbf{a}_1^d | \dots | \mathbf{a}_{t^d}^d)$  and  $\mathbf{c}^d = c_1^d, \dots, c_{t^d}^d$ . Let  $\mathbf{e}^d$  be the noises sampled during encryption.
  2. Set  $\mathbf{x}^{d-1} = (\mathbf{c}^d, 1) \otimes ((\mathbf{s}^d)^{\otimes d}, 1)$ .

Output  $\text{hct}^{\leq D} = \{\text{hct}^d\}_{2 \leq d \leq D}$ .

For every  $1 \leq d \leq D$ ,  $p_d$  is set to be sufficiently large so that  $x^d \in \mathbb{Z}_{p_d}$ . Moreover, observe that for every  $1 \leq d \leq D - 1$ , the number  $t^d$  of elements in  $\mathbf{x}^d$  is a polynomial  $\text{poly}(n)$  multiplicative factor larger than the number  $t^{d+1}$  of elements in  $\mathbf{x}^{d+1}$ . Therefore, every  $\mathbf{x}^d$  contains  $\text{poly}(n)t$  elements, and  $|\text{hct}^{\leq D}| = \text{poly}(n)t$ .

- $\text{DHE.Dec}_{\text{pp}}(\mathbf{s}^{\leq D}, \text{DHE.Eval}_{\text{pp}}(f, \text{hct}^{\leq D}))$ : For a degree  $D$  polynomial  $f$  mapping  $t$  input elements to a single output element, and a *single* ciphertext  $\text{hct}^{\leq D}$  encrypting a length  $t$  vector, the evaluation-decryption equation over the integers is defined by iteratively applying the evaluation-decryption function of HE (see [Claim 6.1](#)) as follows. Set  $f^D = f$ ; in iteration  $d = D, \dots, 1$ , we maintain that

$$f^d(\mathbf{x}^d) + p_d e^{f^d} + q_d o^{f^d} = \text{Pub}^{f^d, \mathbf{A}^d}(\mathbf{c}^d) + \text{Priv}^{f^d, \mathbf{A}^d} \left( (\mathbf{c}^d, 1) \otimes ((\mathbf{s}^d)^{\otimes d}, 1) \right),$$

and define  $f^{d-1} := \text{Priv}^{f^d, \mathbf{A}^d}$  so that

$$f^{d-1}(\mathbf{x}^{d-1}) = \text{Priv}^{f^d, \mathbf{A}^d} \left( (\mathbf{c}^d, 1) \otimes ((\mathbf{s}^d)^{\otimes d}, 1) \right).$$

Summing over all iterations, the overall evaluation-decryption equation is

$$\begin{aligned} \sum_{2 \leq d \leq D} \text{Pub}^{f^d, \mathbf{A}^d}(\mathbf{c}^d) + \text{Priv}^{f^2, \mathbf{A}^2} \left( (\mathbf{c}^2, 1) \otimes ((\mathbf{s}^2)^{\otimes 2}, 1) \right) \\ = f^D(\mathbf{x}) + \sum_{2 \leq d \leq D} \left( p_d e^{f^d} + q_d o^{f^d} \right). \end{aligned} \quad (11)$$

Define  $\text{Pub}^{f, \mathbf{A}^{\leq D}}(\mathbf{c}^{\leq D}) = \sum_{2 \leq d \leq D} \text{Pub}^{f^d, \mathbf{A}^d}(\mathbf{c}^d)$ . We refer to  $\text{Pub}^{f, \mathbf{A}^{\leq D}}$  as the public evaluation-decryption function and  $\text{Priv}^{f^2, \mathbf{A}^2}$  the private evaluation-decryption function of DFE. Importantly,  $\text{Priv}^{f^2, \mathbf{A}^2}$  is a linear function.

From the right-hand side, the output of computing  $f(\mathbf{x})$  over  $\mathbb{Z}_{p_D}$  can be computed:

$$y = \lfloor f(\mathbf{x}) \rfloor_{p_D} = \left\lfloor f^D(\mathbf{x}) + \sum_{2 \leq d \leq D} \left( p_d e^{f^d} + q_d o^{f^d} \right) \right\rfloor_{q_2, p_2, \dots, q_d, p_d, \dots, q_D, p_D},$$

where  $\lfloor \star \rfloor_{q_2, p_2, \dots, q_d, p_d, \dots, q_D, p_D}$  denotes the operation of iteratively modulo  $q_2$  followed by modulo  $p_2$ , followed by  $q_3$ , followed by  $p_3$ , until  $q_D$  followed by  $p_D$ .

Correctness. The correctness of [equation \(11\)](#) follows from the correctness of the evaluation-decryption function of HE in [Claim 6.1](#). To further ensure that  $y = \lfloor f(\mathbf{x}) \rfloor_{p_D}$  is computed correctly, we need to ensure that for all  $2 \leq \rho \leq D$ ,

$$\begin{aligned} p_{\rho-1} &> f^D(\mathbf{x}) + \sum_{\rho \leq d \leq D} \left( p_d e^{f^d} + q_d o^{f^d} \right), \\ q_\rho &> f^D(\mathbf{x}) + \sum_{\rho+1 \leq d \leq D} \left( p_d e^{f^d} + q_d o^{f^d} \right) + p_\rho e^{f^\rho}. \end{aligned}$$

For every  $2 \leq d \leq D$ ,  $f^d$  can be expanded as a sum of monomials  $f^d(\mathbf{x}^d) = \sum_j \text{mnl}_j^d(\mathbf{x}^d) = \prod_\gamma x_\gamma^d$  for  $\gamma \in \Gamma(\text{mnl}_j^d)$ . By [Claim 6.1](#),

$$\begin{aligned} e^{f^d} &= \sum_j e^{\text{mnl}_j^d} \left( \left\{ \langle \mathbf{a}_\gamma^d, \mathbf{s}^d \rangle, x_\gamma^d, e_\gamma^d \right\}_{\gamma \in \Gamma(\text{mnl}_j^d)} \right), \\ o^{f^d} &= \sum_j o^{\text{mnl}_j^d} \left( \left\{ \langle \mathbf{a}_\gamma^d, \mathbf{s}^d \rangle, x_\gamma^d, e_\gamma^d \right\}_{\gamma \in \Gamma(\text{mnl}_j^d)} \right), \end{aligned}$$

and every  $e^{\text{mnl}_j^d}$  is  $\text{poly}(p_d, \Upsilon)$ -bounded, and  $o^{\text{mnl}_j^d}$  is  $\text{poly}(q_d)$ -bounded if  $q_d > n$ . Moreover, if  $s$  is an upper bound on the number of monomials in  $f$ , then  $|f(\mathbf{x})| = \text{poly}(p_D, s)$ . In addition, by our analysis below (see [Claim 6.7](#)), every  $f^d$  contains at most  $m = \text{poly}(n, s)$  monomials.

*Remark 6.6.* [Parameter Setting for Correctness and Robustness] We now describe how to set the public parameters  $p_D, q_D, \dots, p_d, q_d, \dots, p_1$  and parameter of  $\chi$ , which controls its bound  $\Upsilon$ , to satisfy the correctness conditions, and the robustness conditions in [Theorem 6.4](#). First note that  $p_D$  can be arbitrary. Next, to set  $q_D$ , for the correctness conditions, it suffices to set  $q_D$  to a sufficiently large polynomial in  $(\Upsilon, n, p_D, s)$ , and for the robustness conditions, it suffices to ensure that  $\Upsilon = \text{poly}(n, t)$  and  $q_D$  is set to a even larger  $\text{poly}(n, t)$ , where  $t$  is the length of the input  $\mathbf{x}$  encrypted. Overall,  $q_D = \text{poly}(\Upsilon, n, p_D, s, t)$ .

Next, for every  $2 \leq d \leq D$ , to satisfy the correctness condition, we need to set  $p_d$  to be a sufficiently large  $\text{poly}(q_{d+1})$  and  $q_d$  be a sufficiently large  $\text{poly}(p_d)$ . For the robustness conditions, we need to ensure that  $\Upsilon = \text{poly}(n, t^d)$  and  $q_D$  even larger, where  $t^d = \text{poly}(n)t$  is the length of input  $\mathbf{x}^d$  encrypted. Overall, it suffices to have  $p_d = \text{poly}(p_{d+1})$  and  $q_d = \text{poly}(p_d)$ .

Next, we observe several properties of the above construction.

Linear Private Evaluation-Decryption Function. Recall that the evaluation-decryption function over the integers is the sum of a public function  $\text{Pub}^{f, \mathbf{A}^{\leq D}}$  on the ciphertexts  $\mathbf{c}^{\leq D}$  only and a private function  $\text{Priv}^{f^2, \mathbf{A}^2}$  on  $\mathbf{x}^1 = (\mathbf{c}^2, 1) \otimes ((\mathbf{s}^2)^{\otimes 2}, 1)$ , which is *linear*.

Properties of Size and Input-Dependency of  $f$ . In essence, the evaluation-decryption function of DHE iteratively applies the evaluation-decryption function of HE for a sequence of functions  $f^D = f, \dots, f^2$  defined above. We observe that these functions have similar sizes as  $f$  and preserve its input-dependency. First, we bound the number of monomials.

**Claim 6.7.** *Let  $f: \mathbb{Z}^t \rightarrow \mathbb{Z}$  be any degree  $D$  polynomial with at most  $s$  monomials. Then there exists a polynomial  $m$ , such that, for every  $2 \leq d \leq D$ ,  $f^d: \mathbb{Z}^{t^d} \rightarrow \mathbb{Z}$  defined above contains at most  $m(n, s)$  monomials.*

*Proof.* For  $d = D$ , we have  $f^d = f$ , so the claim clearly holds. For every  $d < D$ ,  $f^d(\mathbf{x}^d) = \text{Priv}^{f^{d+1}, \mathbf{A}^{d+1}}((\mathbf{c}^{d+1}, 1) \otimes ((\mathbf{s}^{d+1})^{\otimes d+1}, 1))$ . Its input consists of  $t^d = \text{poly}(n)t^{d+1}$  elements. Recall

that  $\text{Priv}^{f^{d+1}, \mathbf{A}^{d+1}}$  is a sum of functions corresponding to different monomials in  $f^{d+1}$ , that is,  $\text{Priv}^{f^{d+1}, \mathbf{A}^{d+1}}(\mathbf{x}^d) = \sum_j \text{Priv}^{\text{mnl}_j^{d+1}, \mathbf{A}^{d+1}}(\mathbf{x}^d)$ , and each  $\text{Priv}^{\text{mnl}_j^{d+1}, \mathbf{A}^{d+1}}$  has  $\text{poly}(n)$  size. Thus, the total number of monomials  $s^d$  in  $f^d$  is  $\text{poly}(n)s^{d+1}$ . Solving the recursive formula gives that every  $s^d = \text{poly}(n, s)$ . By setting  $m(n, s) = \max_{2 \leq d \leq D} s^d$ , we conclude the claim.  $\square$

Next we consider a special type of functions  $f = (g, G)$  (used later in our construction of FE) that is specified by a function  $g$  and a subset  $G \subseteq [t]$  of indexes of input variables, such that,  $f(\mathbf{x}) = g(x_G)$ . We show that every function  $f^d$  can also be re-written in this form  $f^d = g^d(x_{G^d}^d)$ , where  $G^d$  depends on  $G$ , and  $g^d$  does not. Moreover  $x_{G^d}^d$  depends only on  $x_G$  (and other variables  $\mathbf{s}^{\leq D}, \mathbf{e}^{\leq D}, \mathbf{A}^{\leq D}$  sampled for encryption), and does not depend on other variables  $x_{-G}$  in  $\mathbf{x}$ .

**Claim 6.8** (Preservation of Input Dependency). *Let  $g: \mathbb{Z}^l \rightarrow \mathbb{Z}$  be any degree  $D$  function, and  $G$  any subset of  $[t]$ , and define  $f: \mathbb{Z}^t \rightarrow \mathbb{Z}$  as  $f(\mathbf{x}) = g(x_G)$ . For every function  $f^d: \mathbb{Z}^{t^d} \rightarrow \mathbb{Z}$  with  $2 \leq d \leq D$  and every  $\mathbf{x}^d$  derived from  $\mathbf{x}$  and  $\mathbf{s}^{\leq D}, \mathbf{e}^{\leq D}, \mathbf{A}^{\leq D}$ , there exists a function  $g^d$  depending only on  $(g, \mathbf{A}^{\leq D})$ , and a subset  $G^d \subseteq [t^d]$  depending only on  $G$ , such that,  $f^d(\mathbf{x}^d) = g^d(x_{G^d}^d)$ . Moreover,  $x_{G^d}^d$  can be computed from  $x_G, \mathbf{s}^{\leq D}, \mathbf{e}^{\leq D}, \mathbf{A}^{\leq D}$ .*

*Proof.* We show the claim by induction over  $d$ . In the base case  $d = D$ ,  $f^D(\mathbf{x}) = f(\mathbf{x}) = g(x_G)$ . Clearly,  $g^D = g$  and  $G^D = G$  satisfy the conditions in the claim statement.

Assume that the claim holds for  $d + 1$ , that is, there exists  $g^{d+1}$  depending only on  $(g, \mathbf{A}^{\leq D})$  and  $G^{d+1}$  depending only on  $G$  such that  $f^{d+1}(\mathbf{x}^{d+1}) = g^{d+1}(x_{G^{d+1}}^{d+1})$  and  $x_{G^{d+1}}^{d+1}$  depends only on  $x_G$  (and  $\mathbf{s}^{\leq D}, \mathbf{e}^{\leq D}, \mathbf{A}^{\leq D}$ ). We show that the same holds w.r.t.  $d$  for some  $g^d$  and  $G^d$ . Recall that

$$f^d(\mathbf{x}^d) = \text{Priv}^{f^{d+1}, \mathbf{A}^{d+1}} \left( (c^{d+1}, 1) \otimes ((s^{d+1})^{\otimes d+1}, 1) \right).$$

If  $f^{d+1}$  depends only on  $x_{G^{d+1}}^{d+1}$ , the private evaluation-decryption function  $\text{Priv}^{f^{d+1}, \mathbf{A}^{d+1}}$  only involves ciphertexts encrypting  $x_{G^{d+1}}^{d+1}$ , that is, only  $c_{G^{d+1}}^{d+1}$ . Thus, it can be written as

$$f^d(\mathbf{x}^d) = \text{Priv}^{g^{d+1}, \mathbf{A}^{d+1}} \left( (c_{G^{d+1}}^{d+1}, 1) \otimes ((s^{d+1})^{\otimes d+1}, 1) \right).$$

Therefore, we can define  $g^d := \text{Priv}^{g^{d+1}, \mathbf{A}^{d+1}}$ , which depends only on  $g^{d+1}$  and  $\mathbf{A}^{\leq D}$ , and define  $G^d$  so that  $x_{G^d}^d = (c_{G^{d+1}}^{d+1}, 1) \otimes ((s^{d+1})^{\otimes d+1}, 1)$ , which depends only on  $G^{d+1}$ . Thus,  $f^d(\mathbf{x}^d) = g^d(x_{G^d}^d)$ . By the induction hypothesis, we conclude that  $g^d$  depends only on  $g, \mathbf{A}^{\leq D}$ , and  $G^d$  depends only on  $G$ . Furthermore, since  $x_{G^{d+1}}^{d+1}$  and hence  $c_{G^{d+1}}^{d+1}$  depend only on  $x_G$  (and  $\mathbf{s}^{\leq D}, \mathbf{e}^{\leq D}, \mathbf{A}^{\leq D}$ ), so does  $x_{G^d}^d$ .  $\square$

Local Structure of Noises after Homomorphic Evaluations. For  $f(\mathbf{x}) = g(x_G)$ , based on properties of  $f^d$  shown above, we now analyze the structure of noises derived in the evaluation-decryption function of DHE. We show that every noise  $e^{f^d}$  (or  $o^{f^d}$ ) is a sum of smaller noises  $\sum e^{\text{mnl}_j^d}$  (or  $\sum o^{\text{mnl}_j^d}$ ), each of which can be computed locally from a constant number of variables in  $(x_G, \mathbf{s}^{\leq D}, \mathbf{e}^{\leq D}, \{\mathbf{A}^d \mathbf{s}^d\}_{2 \leq d \leq D})$ . Moreover, this local function itself is independent of  $G$ .

**Claim 6.9** (Additive and Local Structure of Noises). *Let  $g: \mathbb{Z}^l \rightarrow \mathbb{Z}$  be any degree  $D$  function, and  $G$  any subset of  $[t]$ , and define  $f: \mathbb{Z}^t \rightarrow \mathbb{Z}$  as  $f(\mathbf{x}) = g(x_G)$ . For every  $\mathbf{x}$  and  $\mathbf{s}^{\leq D}, \mathbf{e}^{\leq D}, \mathbf{A}^{\leq D}$ , and every  $e^{f^d}$  with  $2 \leq d \leq D$  defined in [equation \(10\)](#), it holds that*

$$e^{f^d} = \sum_j e^{\text{mnl}_j^d} \quad \text{and} \quad \forall d, j, e^{\text{mnl}_j^d} = \tilde{e}^{d,j}(\text{einp}^{d,j}),$$

where  $\tilde{e}^{d,j}$  is a function independent of  $G$ , and its input  $\text{inp}^{d,j} \subseteq \{x_G, \mathbf{s}^{\leq D}, \mathbf{e}^{\leq D}, \{\mathbf{A}^d \mathbf{s}^d\}_{2 \leq d \leq D}\}$  is a subset of a constant number of variables satisfying that, for every  $c_t^d$ , if some variable in  $\mathbf{a}_t^d$  or  $e_t^d$  or  $\langle \mathbf{a}_t^d, \mathbf{s}^d \rangle$  belongs to  $\text{inp}^{d,j}$ , then  $c_t^d$  can be computed from  $\text{inp}^{d,j}$ .

In addition, the same holds for every  $o^d$  for  $2 \leq d \leq D$  w.r.t. some  $\{\tilde{o}^{d,j}, \text{oinp}^{d,j}\}$ .

*Proof.* We prove the claim w.r.t.  $e^f$  and  $e^{\text{mnl}_j^d}$ 's; the proof for  $o^f$  and  $o^{\text{mnl}_j^d}$ 's follows the same way.

By [Claim 6.8](#), when  $f(\mathbf{x}) = g(x_G)$ , every  $f^d(\mathbf{x}^d) = g^d(x_{G^d}^d)$ , where  $g^d$  is independent of  $G$ . The local structure of noises of HE ([Claim 6.1](#)) implies that  $e^{\text{mnl}_j^d}$  depends only on variables in  $\mathbf{x}^d$  that the monomial  $\text{mnl}_j^d$  in  $g^d$  depends on, that is,

$$e^{\text{mnl}_j^d} = e^{\text{mnl}_j^d} \left( \left\{ \langle \mathbf{a}_\gamma^d, \mathbf{s}^d \rangle, x_\gamma^d, e_\gamma^d \right\}_{\gamma \in \Gamma(\text{mnl}_j^d)} \right),$$

where the computation of  $e^{\text{mnl}_j^d}$  depends only on  $\text{mnl}_j^d$  and hence is also independent of  $G$ .

Since  $\mathbf{x}^d = (\mathbf{c}^{d+1}, 1) \otimes ((\mathbf{s}^{d+1})^{\otimes d+1}, 1)$ , every  $x_\gamma^d$  depends on at most  $d+1$  variables  $s_l^{d+1}$ , and a single variable  $c_l^{d+1} = \lfloor \langle \mathbf{a}_l^{d+1}, \mathbf{s}^{d+1} \rangle + x_l^{d+1} + p_{d+1} e_l^d \rfloor_{p_d}$ . Thus,  $e^{\text{mnl}_j^d}$  can in turn be computed from  $\{\langle \mathbf{a}_\gamma^d, \mathbf{s}^d \rangle, e_\gamma^d\}_{\gamma \in \Gamma(\text{mnl}_j^d)}$ , and variables  $\{s_l^{d+1}\} \cup \{\langle \mathbf{a}_l^{d+1}, \mathbf{s}^{d+1} \rangle, x_l^{d+1}, e_l^d\}$  that determine  $x_\gamma^d$  for all  $\gamma \in \Gamma(\text{mnl}_j^d)$  (by first recomputing  $x_\gamma^d$  from the latter and then computing  $e^{\text{mnl}_j^d}$  as before). Clearly the function computed is still independent of  $G$ , and the number of input variables now increases by a constant multiplicative factor.

Repeat the above to recursively replace variables  $x_\gamma^\rho$  for  $\rho = d+1, \dots, D$  as above, until only variables  $x_\gamma^D = x_\gamma$  are used. We obtain that  $e^{\text{mnl}_j^d}$  can be computed using a function independent of  $G$  from a subset of variables  $\text{inp}^{d,j} \subseteq \{\mathbf{x}, \mathbf{s}^{\leq D}, \mathbf{e}^{\leq D}, \{\mathbf{A}^d \mathbf{s}^d\}_{2 \leq d \leq D}\}$  that has constant size. Since, in each recursive iteration, variables  $\langle \mathbf{a}_l^d, \mathbf{s}^d \rangle, x_l^d, e_l^d$  related to a ciphertext  $c_l^d$  are always added together to the set of relevant variables, and later  $x_l^d$  is recursively replaced with variables sufficient for computing it, we conclude that for every  $c_t^d$ , if  $e_t^d$  or  $\langle \mathbf{a}_t^d, \mathbf{s}^d \rangle$  belongs to  $\text{inp}^{d,j}$ , then  $c_t^d$  can be computed from  $\text{inp}^{d,j}$ . Thus, the condition in the claim statement also holds, as  $\text{inp}^{d,j}$  does not include variables in  $\mathbf{a}_t^d$ . Finally, since in each recursive iteration, the number of relevant variables increase by a constant factor, the total number of input variables after at most  $D$  iterations is still a constant.

Finally, we argue that  $\text{inp}^{d,j}$  contain only  $x_G$  and no other variables in  $\mathbf{x}$ . By [Claim 6.8](#), the input  $x_{G^d}^d$  of  $g^d(x_{G^d}^d)$  depends only on  $x_G$ . Since  $e^{\text{mnl}_j^d}$  depends on at most variables  $x_{G^d}^d$  in  $\mathbf{x}^d$ , it in turn depends on at most variables  $x_G$ .  $\square$

### 6.3 Construction of DFE

For arbitrary positive integers  $D$ , and arbitrary polynomials  $p_D$  and  $\ell$  in  $\lambda$ , we now construct a secret key FE scheme  $\text{DFE}^{N,S}$  for computing over  $\mathbb{Z}_{p_D}$  the following family  $\text{DF}^{N,S}$  of polynomials.

- Family  $\text{DF}^{N,S} = \{\text{DF}_\lambda^{N,S}\}$  indexed by arbitrary polynomials  $N, S$ , contains locality  $\ell = \ell(\lambda)$  degree  $D$  polynomials  $f: \mathbb{Z}_{p_D}^N \rightarrow \mathbb{Z}_{p_D}^M$  of size  $S = S(\lambda)$ , mapping inputs of length  $N = N(\lambda)$  to outputs of length  $M(\lambda) \leq S(\lambda)$ . We write  $f = \{f_i(x)\}_{i \in [M]}$  to mean that the  $i$ 'th output element is computed by the polynomial  $f_i$ .

We construct  $\text{DFE} = \text{DFE}^{N,S}$  for computing the above family using the following tools.

- The HE scheme DHE for computing degree  $D$  polynomials constructed in [Section 6.2](#). The public parameters  $\mathbf{pp} = (p_D, q_D, \dots, p_1, \chi)$  satisfy similar conditions described in the construction of DHE (see [Remark 6.6](#)). In particular, the noise distribution is polynomially  $\Upsilon$ -bounded, and all moduli  $p_d, q_d$  are polynomially large. We detail on the setting of the parameters at the end of the correctness analysis of DFE.
- A collection of  $\eta$ -noisy secret-key FE schemes  $\{\text{NFE}^{N',S}\}$  for computing linear functions  $\mathbb{Z}_Q^{N'} \rightarrow \mathbb{Z}_Q^S$  for some super-polynomial  $Q$  and  $N' = \text{poly}(\lambda)N$ , with weak correctness, 1-key  $O(\mu)$ -Sel-Sim security, and special-purpose  $(1 - \alpha)$ -sublinear compactness, i.e., with ciphertexts of size  $\text{poly}(\lambda)(N + S^{1-\alpha})$ . We need the distribution  $\eta$  to be of the form

$$p_D \Phi + \sum_{2 \leq d \leq D} \left( p_d \cdot \sum_{j \in [m]} \Phi^{d,j} + q_d \cdot \sum_{j \in [m]} \Psi^{d,j} \right),$$

where  $\Phi, \Phi^{d,1} \parallel \dots \parallel \Phi^{d,m}$ , and  $\Psi^{d,1} \parallel \dots \parallel \Psi^{d,m}$  are sampled from distributions  $\mathcal{Z}_{\bar{p}_D}, \mathcal{Z}_{p_d}$ , and  $\mathcal{Z}_{q_d}$ , respectively, and  $m$  is the value from [Claim 6.7](#). The distribution  $\mathcal{Z}_{\bar{p}_D}$  is over  $\mathbb{Z}^S$ , and  $\mathcal{Z}_{p_d}$  and  $\mathcal{Z}_{q_d}$  are distributions over  $\mathbb{Z}^{m \cdot S}$ . Moreover, the distributions  $\mathcal{Z}_{\bar{p}_D}, \mathcal{Z}_{p_d}$ , and  $\mathcal{Z}_{q_d}$  are  $(\lambda^{\varepsilon_2}, \mu)$ -flawed-smudging for  $B_{\bar{p}_D}$ -bounded,  $B_{p_d}$ -bounded, and  $B_{q_d}$ -bounded distributions, respectively, for some constant  $\varepsilon_2 \in (0, 1)$  and polynomial bounds  $B_{\bar{p}_D}, B_{p_d}$ , and  $B_{q_d}$ , which are set below. We further require the distributions  $\mathcal{Z}_\star$  to be  $\text{poly}(B_\star, \lambda)$ -bounded.

**Construction of DFE<sup>N,S</sup>.** Using the above tools, our FE scheme DFE consists of the following algorithms:

- $\text{DFE.Setup}(1^\lambda)$ , on input the security parameter, generates a master secret key  $\text{nmsk} \leftarrow \text{NFE.Setup}(1^\lambda)$  for the noisy FE scheme, and samples  $\mathbf{A}^{\leq D} = \{\mathbf{A}^D, \dots, \mathbf{A}^2\}$  randomly. Output  $\text{msk} = (\text{nmsk}, \mathbf{A}^{\leq D})$ .

*We emphasize that the random matrices  $\mathbf{A}^d$  used in DHE encryption are sampled by the setup algorithm, and reused in different encryptions.*

- $\text{DFE.Enc}(\text{msk} = (\text{nmsk}, \mathbf{A}^{\leq D}), \mathbf{x})$ , on input  $x \in \mathbb{Z}_{p_D}^N$ , does:
  - Sample  $\mathbf{s}^{\leq D} \leftarrow \text{DHE.KeyGen}_{\mathbf{pp}}(1^n)$ , where  $n$  is a sufficiently large polynomial in  $\lambda$ , (and  $\mathbf{pp} = (p_D, \dots, p_1)$  is as described above).
  - Encrypt  $\text{hct}^{\leq D} \leftarrow \text{DHE.Enc}_{\mathbf{pp}}(\mathbf{s}^{\leq D}, \mathbf{x}; \mathbf{A}^{\leq D}, \mathbf{e}^{\leq D})$ , using the matrices  $\mathbf{A}^{\leq D}$  contained in the master secret key, and freshly sampled noises  $\mathbf{e}^{\leq D}$ .
  - Generate  $\mathbf{x}^1 = (\mathbf{c}^2, 1) \otimes ((\mathbf{s}^2)^{\otimes 2}, 1)$ .
  - Encrypt  $\text{nct} \leftarrow \text{NFE.Enc}(\text{nmsk}, \mathbf{x}^1)$ .

Output  $\text{Dct} = (\text{hct}^{\leq D}, \text{nct})$ .

- $\text{DFE.KeyGen}(\text{msk} = (\text{nmsk}, \mathbf{A}^{\leq D}), f)$ , on input a degree  $D$  polynomial  $f \in \text{DF}^{N,S}$ , does the following:
  - For every  $i \in [M]$ , let  $f_i^D = f_i, f_i^{D-1}, \dots, f_i^2$  be the sequence of functions that the evaluation-decryption function of DHE when applied to  $f_i$  generates. They form

for each  $d$ ,  $f^d = \{f_i^d\}$ . Since each  $f$  has locality  $\ell$ , each  $f_i$  contains at most  $s = \text{poly}(\lambda, \ell)$  number of monomials. By [Claim 6.7](#), every component  $f_i^d$  has at most  $m = m(n, s)$  monomials. Recall that  $\text{Priv}^{f_i^d, \mathbf{A}^2}$  is a linear function. Let  $\text{Priv}^{f^2, \mathbf{A}^2} := \{\text{Priv}^{f_i^d, \mathbf{A}^2}\}_{i \in [M]}$ .

– Generate the key  $\text{nsk} \leftarrow \text{NFE.KeyGen}(\text{nmsk}, \text{Priv}^{f^2, \mathbf{A}^2})$ .<sup>13</sup>

Output  $\text{Dsk} = \text{nsk}$ .

- $\text{DFE.Dec}(\text{Dsk}, \text{Dct})$  on input  $\text{Dsk} = \text{nsk}$  and  $\text{Dct} = (\text{hct}^{\leq D}, \text{nct})$ , compute for every  $2 \leq d \leq D$  and for all  $i$ ,  $\text{pub}_i^d = \text{Pub}^{f_i^d, \mathbf{A}^d}(\mathbf{c}^d)$ . Let  $\text{pub}_i := \sum_{2 \leq d \leq D} \text{pub}_i^d$ ,  $\text{pub} = \{\text{pub}_i\}$ , and let  $\mathcal{R}_i := [-\tilde{B} - \text{pub}_i, \tilde{B} - \text{pub}_i]$ , where  $\tilde{B}$  is the polynomial bound derived in the correctness analysis below. Then compute  $\mathbf{z} = \text{NFE.Dec}(\text{nsk}, \text{nct}, \mathcal{R}_1, \dots, \mathcal{R}_M)$  and let  $\mathbf{z}' := \mathbf{z} + \text{pub}$ . Finally, the output of computing  $f(\mathbf{x})$  over  $\mathbb{Z}_{p_D}$  can be computed from  $\mathbf{z}'$  as

$$\mathbf{y} = \lfloor f(\mathbf{x}) \rfloor_{p_D} = \lfloor \mathbf{z}' \rfloor_{q_2, p_2, \dots, q_d, p_d, \dots, q_D, p_D}.$$

Correctness: The evaluation-decryption function of DHE (see [equation \(11\)](#)) implies that for every  $i$ ,

$$\sum_{2 \leq d \leq D} \text{Pub}^{f_i^d, \mathbf{A}^d}(\mathbf{c}^d) + \text{Priv}^{f_i^d, \mathbf{A}^2}(\mathbf{x}^1) = \lfloor f_i(\mathbf{x}) \rfloor_{p_D} + p_D o_i^{f_i} + \sum_{2 \leq d \leq D} \left( p_d e_i^{f_i^d} + q_d o_i^{f_i^d} \right), \quad (12)$$

where

$$o_i^{f_i} = \left\lfloor \frac{f_i(\mathbf{x})}{p_D} \right\rfloor, \quad e_i^{f_i^d} = \sum_{j \in [m]} e^{\text{mnl}_{i,j}^d}, \quad o_i^{f_i^d} = \sum_{j \in [m]} o^{\text{mnl}_{i,j}^d},$$

and  $\text{mnl}_{i,j}^d$  is one of the (at most)  $m$  monomials of  $f_i^d$ . In addition, every  $e^{\text{mnl}_{i,j}^d}$  is  $\text{poly}(\Upsilon, p_d)$  bounded, and every  $o^{\text{mnl}_{i,j}^d}$  is  $\text{poly}(q_d)$  bounded. Therefore, all  $\text{pub}_i + \text{Priv}^{f_i^d, \mathbf{A}^2}(\mathbf{x}^1)$  are bounded by some polynomial bound  $\tilde{B}$ . This implies that  $\text{Priv}^{f_i^d, \mathbf{A}^2}(\mathbf{x}^1) \in [-\tilde{B} - \text{pub}_i, \tilde{B} - \text{pub}_i] = \mathcal{R}_i$  for all  $i$ . Hence, the weak correctness of NFE implies that

$$\mathbf{z} = \lfloor \text{Priv}^{f^2, \mathbf{A}^2}(\mathbf{x}^1) + \mathbf{\Delta} \rfloor_Q,$$

for some  $\mathbf{\Delta}$  in the support of  $\eta$ . Therefore,

$$\mathbf{z}' = \lfloor \text{pub} + \text{Priv}^{f^2, \mathbf{A}^2}(\mathbf{x}^1) + \mathbf{\Delta} \rfloor_Q.$$

We rename  $o_i^f = o_i^{f_i}$ ,  $e_i^{f_i^d} = e_i^{f_i^d}$ ,  $o_i^{f_i^d} = o_i^{f_i^d}$ ,  $e_i^{d,j} = e^{\text{mnl}_{i,j}^d}$ ,  $o_i^{d,j} = o^{\text{mnl}_{i,j}^d}$ , so that we can write [equation \(12\)](#) in vector form and obtain

$$\mathbf{z}' = \left\lfloor \left[ \lfloor f(\mathbf{x}) \rfloor_{p_D} + p_D \mathbf{o}^f + \sum_{2 \leq d \leq D} \left( p_d \sum_{j \in [m]} \mathbf{e}^{d,j} + q_d \sum_{j \in [m]} \mathbf{o}^{d,j} \right) + \mathbf{\Delta} \right] \right\rfloor_Q.$$

<sup>13</sup>The output length of  $\text{Priv}^{f^2, \mathbf{A}^2}$  equals  $M$ , i.e., the output length of  $f$ , whereas NFE supports functions with output length  $S$ . The value  $M$  might be smaller than  $S$ , in which case we can pad  $\text{Priv}^{f^2, \mathbf{A}^2}$  with additional 0 output elements.

By the definition of  $\eta$ ,  $\Delta$  is of the form

$$\Delta = p_D \Phi + \sum_{2 \leq d \leq D} \left( p_d \cdot \sum_{j \in [m]} \Phi^{d,j} + q_d \cdot \sum_{j \in [m]} \Psi^{d,j} \right),$$

where  $\Phi$ ,  $\Phi^{d,1} \parallel \dots \parallel \Phi^{d,m}$ , and  $\Psi^{d,1} \parallel \dots \parallel \Psi^{d,m}$  are in the support of  $\mathcal{Z}_{\bar{p}_D}$ ,  $\mathcal{Z}_{p_d}$ , and  $\mathcal{Z}_{q_d}$ , respectively. We thus have

$$\mathbf{z}' = \left[ \lfloor f(\mathbf{x}) \rfloor_{P_D} + p_D(\mathbf{o}^f + \Phi) + \sum_{2 \leq d \leq D} \left( p_d \sum_{j \in [m]} (\mathbf{e}^{d,j} + \Phi^{d,j}) + q_d \sum_{j \in [m]} (\mathbf{o}^{d,j} + \Psi^{d,j}) \right) \right]_Q. \quad (13)$$

The  $\mathbf{e}^{d,j}$ ,  $\mathbf{o}^{d,j}$ , and  $\mathbf{o}^f$  are  $B_{p_d} = \text{poly}(\Upsilon, p_d)$ ,  $B_{q_d} = \text{poly}(q_d)$ , and  $B_{\bar{p}_D} = \text{poly}(p_D, s) < B_{p_D}$  bounded, respectively. Moreover, every  $\Phi^{d,j}$ ,  $\Psi^{d,j}$  and  $\Phi$  is  $\text{poly}(B_{p_d}, \lambda)$ ,  $\text{poly}(B_{q_d}, \lambda)$ , and  $\text{poly}(B_{\bar{p}_D}, \lambda)$  bounded, respectively. Since the modulus  $Q$  is super-polynomial, the modular operation can be removed from the above equation.

To ensure that  $\mathbf{y} = \lfloor f(\mathbf{x}) \rfloor_{p_D}$  can be computed from  $\mathbf{z}'$  via iteratively reducing modulo  $q_2, p_2, \dots, q_d, p_d, \dots, q_D, p_D$ , we need that for every  $2 \leq \rho \leq D$ ,

$$\lfloor f(\mathbf{x}) \rfloor_{P_D} + p_D(\mathbf{o}^f + \Phi) + \sum_{\rho+1 \leq d \leq D} p_d \sum_{j \in [m]} (\mathbf{e}^{d,j} + \Phi^{d,j}) + \sum_{\rho+1 \leq d \leq D} q_d \sum_{j \in [m]} (\mathbf{o}^{d,j} + \Psi^{d,j})$$

is  $q_\rho$  bounded, and

$$\lfloor f(\mathbf{x}) \rfloor_{P_D} + p_D(\mathbf{o}^f + \Phi) + \sum_{\rho+1 \leq d \leq D} \left( p_d \sum_{j \in [m]} (\mathbf{e}^{d,j} + \Phi^{d,j}) + q_d \sum_{j \in [m]} (\mathbf{o}^{d,j} + \Psi^{d,j}) \right)$$

is  $p_\rho$  bounded. Since  $\Phi^{d,j}$ ,  $\Psi^{d,j}$  and  $\Phi$  are respectively polynomial in the magnitude of  $\mathbf{e}^{d,j}$ ,  $\mathbf{o}^{d,j}$  and  $\mathbf{o}^f$ , the above conditions can be satisfied by setting parameters as in [Remark 6.6](#). That is,  $\Upsilon = \text{poly}(n, t = N)$ ,  $q_D = \text{poly}(\Upsilon, n, p_D, s, t = N)$ ,  $p_d = \text{poly}(q_{d+1})$  and  $q_d = \text{poly}(p_d)$ , where  $t = N$  is the length of the input string  $\mathbf{x}$ .

*Special-purpose sublinear compactness:* By the efficiency of the DHE scheme, the size of its ciphertext is

$$|\text{hct}^{\leq D}| = \text{poly}(n)N = \text{poly}(\lambda)N,$$

and the size of  $\mathbf{x}^1 = (\mathbf{c}^2, 1) \otimes ((\mathbf{s}^2)^{\otimes 2}, 1)$  is also  $\text{poly}(\lambda)N$ . The special-purpose sublinear compactness of NFE further implies that

$$|\text{nct}| \leq \text{poly}(\lambda)(N + S^{1-\alpha}).$$

Therefore the overall size of a ciphertext of DFE is

$$|\text{ct}| \leq \text{poly}(\lambda)(N + S^{1-\alpha}).$$

**Special-purpose simulation security.** We show that our DFE scheme satisfies a weak version of (1-key) simulation security. Roughly speaking, it guarantees that a secret key  $\text{sk}$  of a function  $f$  and a ciphertext  $\text{ct}$  of an input  $\mathbf{x}$  of DFE can be simulated by a universal simulator  $\text{Sim}$ , using the output  $\mathbf{y} = f(\mathbf{x})$ , together with some ‘‘leakage’’ of a bounded  $O(\lambda^{\varepsilon_2})$  number of variables in  $\mathbf{x}$ . More precisely, since DFE is a secret key encryption scheme, we need to directly consider the distribution of  $\text{sk}$  and many ciphertexts  $\text{ct}$ , and  $\text{ct}_1, \dots, \text{ct}_t$ , and the simulation additionally simulates  $\text{ct}_i$ ’s using the input  $\mathbf{x}_i$  encrypted inside.

**Definition 6.10.** We say that  $\text{DFE}^{N,S}$  satisfies special-purpose  $\mu$ -simulation security if there exists an efficient universal simulator  $\text{Sim}$  such that, for

- every distribution  $\{\mathcal{FN}\}_\lambda$  over  $f \in \mathcal{DF}^{N,S}$ ,
- every distribution  $\{\mathcal{X}\}_\lambda$  over  $\mathbf{x}$  in  $\mathbb{Z}_{p_D}^N$ ,
- every sequence of vectors  $\{\mathbf{x}_1, \dots, \mathbf{x}_t\}$  where  $\mathbf{x}_i \in \mathbb{Z}_{q_D}^N$ ,

there exist correlated random variables  $(x_K, K, \text{st})$  sampled (potentially inefficiently) by  $\mathcal{D}_{\text{Sim}}$ , such that the following distributions are  $\mu$ -indistinguishable,

$$\begin{aligned} \text{Real} &= \left\{ \begin{array}{l} f \leftarrow \mathcal{FN}, \mathbf{x} \leftarrow \mathcal{X} \\ \text{msk} \leftarrow \text{DFE.Setup}(1^\lambda) \\ \text{sk} \leftarrow \text{DFE.KeyGen}(\text{msk}, f) \\ \text{ct} \leftarrow \text{DFE.Enc}(\text{msk}, \mathbf{x}) \\ \{\text{ct}_i \leftarrow \text{DFE.Enc}(\text{msk}, \mathbf{x}_i)\}_{i \in [t]} \end{array} : f, \mathbf{x}, (\text{sk}, \text{ct}, \text{ct}_1, \dots, \text{ct}_t) \right\}_{\lambda \in \mathbb{N}}, \\ \text{Ideal} &= \left\{ \begin{array}{l} f \leftarrow \mathcal{FN}, \\ (x_K, K, \text{st}) \leftarrow \mathcal{D}_{\text{Sim}}(f), \\ \bar{\mathbf{x}} \leftarrow \mathcal{X}|_{x_K, K}, \end{array} : f, \bar{\mathbf{x}}, \text{Sim}(\text{st}, f, y = f(\bar{\mathbf{x}}), \mathbf{x}_1, \dots, \mathbf{x}_t) \right\}_{\lambda \in \mathbb{N}}, \end{aligned}$$

and with probability at least  $1 - \mu$ ,  $|K| \leq O(\lambda^{\varepsilon_2} \ell)$  for some constant  $\varepsilon_2 \in (0, 1)$ , where  $\ell$  is the maximum locality of  $f \in \mathcal{DF}^{N,S}$ .

We next show that our FE scheme DFE satisfies this special-purpose simulation security.

**Lemma 6.11.** *Assume that DHE has  $\mu$ -robustness for secrets with  $(1 - o(1))n$  min-entropy, for all polynomials  $N'$  and  $S$ ,  $\text{NFE}^{N',S}$  satisfies 1-key  $O(\mu)$ -Sel-Sim security. Then, for all polynomials  $N$  and  $S$ ,  $\text{DFE}^{N,S}$  satisfies special-purpose  $O(\mu)$ -simulation security.*

*Proof.* We construct  $\text{Sim}$  and  $\mathcal{D}_{\text{Sim}}$  and show that the corresponding ideal distribution  $\text{Ideal}$  is  $O(\mu)$ -indistinguishable to  $\text{Real}$  via a sequence of hybrids  $H_0, \dots, H_4$ , where  $H_0 = \text{Real}$  and  $H_4 = \text{Ideal}$ . We define  $\text{Sim}$  and  $\mathcal{D}_{\text{Sim}}$  in the description of  $H_4$ .

**Hybrid  $H_0$**  is exactly the  $\text{Real}$  distribution. By the construction of DFE, the secret key  $\text{sk}$  for a function  $f$  is a secret key  $\text{nsk}$  of the NFE scheme for the function  $\text{Priv}^{f^2, \mathbf{A}^2}$ , and a ciphertext  $\text{ct} = (\text{hct}^{\leq D}, \text{nct})$  consists of  $\text{hct}^{\leq D} = (\mathbf{A}^{\leq D}, \mathbf{c}^{\leq D})$  encrypting  $\mathbf{x}$ , and  $\text{nct}$  encrypts  $\mathbf{x}^1$ . Therefore, the distribution of  $\text{Real}$  is

$$\left\{ f, \mathbf{x}, \text{sk} = \text{nsk}, \left( (\mathbf{A}^{\leq D}, \mathbf{c}^{\leq D}), \text{nct} \right), \left\{ \left( (\mathbf{A}^{\leq D}, \mathbf{c}_i^{\leq D}), \text{nct}_i \right) \right\}_{i \in [t]} \right\},$$

where  $f \leftarrow \mathcal{FN}$ ,  $\mathbf{x} \leftarrow \mathcal{X}$ , and all keys and ciphertexts are generated honestly. Note that the matrices  $\mathbf{A}^{\leq D}$  are shared between different ciphertexts of DFE.

**Hybrid  $H_1$**  is the same as  $H_0$ , except that the secret key  $\text{nsk}$  and the ciphertexts  $\text{nct}$  and  $\text{nct}_1, \dots, \text{nct}_t$  of the NFE scheme NFE are now simulated using the simulator  $\text{NSim}$ . This yields the following distribution (with items re-ordered).

$$\left\{ f, \mathbf{x}, (\mathbf{A}^{\leq D}, \mathbf{c}^{\leq D}), \left\{ \left( (\mathbf{A}^{\leq D}, \mathbf{c}_i^{\leq D}) \right) \right\}_{i \in [t]}, \left( \widetilde{\text{nsk}}, \widetilde{\text{nct}}, \{\widetilde{\text{nct}}_i\} \right) \leftarrow \text{NSim}(\text{Priv}^{f^2, \mathbf{A}^2}, Y = \lfloor \text{Priv}^{f^2, \mathbf{A}^2}(\mathbf{x}^1) + \mathbf{\Delta} \rfloor_Q, \mathbf{x}_1^1, \dots, \mathbf{x}_t^1) \right\},$$

where  $\Delta$  is sampled from the distribution  $\eta$ , and  $\mathbf{x}^1$  and  $\mathbf{x}_1^1, \dots, \mathbf{x}_t^1$  are the input vectors encrypted using NFE in each encryption of DFE.

The  $O(\mu)$ -Sel-Sim-security of NFE implies that  $H_0$  and  $H_1$  are  $O(\mu)$ -indistinguishable.

In the correctness analysis, we derived that the following equation holds (see [equation \(13\)](#)):

$$\begin{aligned} & \sum_{2 \leq d \leq D} \text{Pub}^{f^d, \mathbf{A}^d}(\mathbf{c}^d) + \text{Priv}^{f^2, \mathbf{A}^2}(\mathbf{x}^1) + \Delta \\ &= \lfloor f(\mathbf{x}) \rfloor_{P_D} + p_D(\mathbf{o}^f + \Phi) + \sum_{2 \leq d \leq D} \left( p_d \sum_{j \in [m]} (\mathbf{e}^{d,j} + \Phi^{d,j}) + q_d \sum_{j \in [m]} (\mathbf{o}^{d,j} + \Psi^{d,j}) \right), \end{aligned}$$

where  $\Phi$ ,  $\Phi^{d,1} \parallel \dots \parallel \Phi^{d,m}$ , and  $\Psi^{d,1} \parallel \dots \parallel \Psi^{d,m}$  are sampled from the distributions  $\mathcal{Z}_{\bar{p}_D}$ ,  $\mathcal{Z}_{p_d}$ , and  $\mathcal{Z}_{q_d}$ , respectively. Therefore,

$$\begin{aligned} Y = \lfloor \text{Priv}^{f^2, \mathbf{A}^2}(\mathbf{x}^1) + \Delta \rfloor_Q &= \left[ \lfloor f(\mathbf{x}) \rfloor_{P_D} - \sum_{2 \leq d \leq D} \text{Pub}^{f^d, \mathbf{A}^d}(\mathbf{c}^d) \right. \\ & \left. + p_D(\mathbf{o}^f + \Phi) + \sum_{2 \leq d \leq D} \left( p_d \sum_{j \in [m]} (\mathbf{e}^{d,j} + \Phi^{d,j}) + q_d \sum_{j \in [m]} (\mathbf{o}^{d,j} + \Psi^{d,j}) \right) \right]_Q. \end{aligned}$$

**Hybrid  $H_2$**  is the same as  $H_1$ , except that we apply the property of  $(\lambda^{\varepsilon_2}, \mu)$ -flawed-smudging distributions  $\mathcal{Z}_{\bar{p}_D}$ ,  $\mathcal{Z}_{p_d}$ , and  $\mathcal{Z}_{q_d}$  to argue that the sums of noises and samples from these distributions hide the noises at most locations, which means only a few variables related to the generation of HE ciphertexts  $V = (\mathbf{x}, \mathbf{A}^{\leq D}, \mathbf{s}^{\leq D}, \mathbf{e}^{\leq D}, \{\mathbf{A}^d \mathbf{s}^d\})$  are “compromised”.

We will use [Lemma 3.18](#), which states that for noises generated by  $E(V)$ , where  $E$  is a function and  $V \leftarrow \mathcal{V}$  is a sample from a distribution, adding  $E(V)$  with a sample  $Z \leftarrow \mathcal{Z}$  from the flawed-smudging distribution can hide most locations of  $E(V)$  and in turn  $V$ . The set of “compromised” noises  $E_{\text{bad}}$  is chosen by a collection of randomized predicates  $\text{BAD}_\rho$  —  $E_\rho(V)$  is “compromised” if  $\text{bad}_\rho \leftarrow \text{BAD}_\rho(E_\rho(V), Z)$  happens to be 1 — and the set of “compromised” variables  $V_{\Gamma(E_{\text{bad}})}$  simply consists of those that  $E_{\text{bad}}$  depends on. All other variables  $V_{-\Gamma(E_{\text{bad}})}$  are hidden. This is formalized as that given  $E(V) + Z$ ,  $V$  is distributed identically to a new sampling  $\bar{V} \leftarrow \mathcal{V}$  with only  $V_{\Gamma(E_{\text{bad}})}$  fixed. More precisely, the following distributions are identical:

$$\begin{aligned} \mathcal{D}_1 &= \left\{ \begin{array}{l} V \leftarrow \mathcal{V}, Z \leftarrow \mathcal{Z} \\ \text{bad} \leftarrow \{\text{BAD}_\rho(E_\rho(V), Z)\}_\rho \end{array} : (V, E(V) + Z, \text{bad}) \right\}, \\ \mathcal{D}_2 &= \left\{ \begin{array}{l} V \leftarrow \mathcal{V}, Z \leftarrow \mathcal{Z} \\ \text{bad} \leftarrow \{\text{BAD}_\rho(E_\rho(V), Z)\}_\rho \\ I = \Gamma(E_{\text{bad}}), \bar{V} \leftarrow \mathcal{V}|_{V_I, I} \end{array} : (\bar{V}, E(V) + Z, \text{bad}) \right\}. \end{aligned}$$

To apply this property, we need to specify the relevant  $\mathcal{V}$ ,  $E$ , and  $\mathcal{Z}$ . In  $H_1$ , the noises are functions of variables used for producing the DHE ciphertexts. Therefore,

$$\mathcal{V} : \mathbf{x} \leftarrow \mathcal{X}, \text{ sample } \mathbf{A}^{\leq D}, \mathbf{s}^{\leq D}, \mathbf{e}^{\leq D}; \text{ output } V = (\mathbf{x}, \mathbf{A}^{\leq D}, \mathbf{s}^{\leq D}, \mathbf{e}^{\leq D}, \{\mathbf{A}^d \mathbf{s}^d\}).$$

Then let  $E = E^f$  be the function that computes all noises from  $V$ , which depends on  $f$ :

$$E^f(V) := \left( \left\{ e_i^{d,j} = \tilde{e}_i^{d,j}(\text{einp}_i^{d,j}) \right\}_{d,j,i}, \left\{ o_i^{d,j} = \tilde{o}_i^{d,j}(\text{oinp}_i^{d,j}) \right\}_{d,j,i}, \left\{ o_i^f = \tilde{o}_i^f(\text{oinp}_i^f) \right\}_i \right).$$

Above every  $e_i^{d,j}, o_i^{d,j}$  is computed from variables in  $\text{einp}_i^{d,j}, \text{oinp}_i^{d,j}$  defined in [Claim 6.9](#), each of which contains only a *constant* number of variables in  $V$ . In addition, every  $\tilde{o}_i^f(\text{oinp}_i^f)$  computes  $o_i^f = \lfloor f_i(\mathbf{x})/p_D \rfloor$ . Since  $f$  has locality  $\ell$ ,  $\text{oinp}_i^f$  contains at most  $\ell$  variables  $x$ . Therefore, for every subset  $E_{\text{bad}}$  of compromised noises, the subset  $I = \Gamma(E_{\text{bad}})$  of compromised input variables that  $E_{\text{bad}}$  depends on contains at most  $O(|\text{bad}|_1)$  variables in  $(\mathbf{A}^{\leq D}, \mathbf{s}^{\leq D}, \mathbf{e}^{\leq D}, \{\mathbf{A}^d \mathbf{s}^d\})$  and at most  $O(\ell|\text{bad}|_1)$  variables in  $\mathbf{x}$ . We have

$$I = \Gamma(E_{\text{bad}}) = \left( \bigcup_{e_i^{d,j} \in E_{\text{bad}}} \text{einp}_i^{d,j} \right) \cup \left( \bigcup_{o_i^{d,j} \in E_{\text{bad}}} \text{oinp}_i^{d,j} \right) \cup \left( \bigcup_{o_i^f \in E_{\text{bad}}} \text{oinp}_i^f \right).$$

Finally, in  $H_1$  the smudging noises are sampled from the product of different flawed-smudging distributions:

$$\begin{aligned} \mathcal{Z} : & \left( \{\Phi^{d,1} \parallel \dots \parallel \Phi^{d,m} \leftarrow \mathcal{Z}_{p_d}\}_d, \{\Psi^{d,1} \parallel \dots \parallel \Psi^{d,m} \leftarrow \mathcal{Z}_{q_d}\}_d, \Phi \leftarrow \mathcal{Z}_{\bar{p}_D} \right); \\ & \text{output } Z = \{\Phi^{d,j}, \Psi^{d,j}, \Phi\}. \end{aligned}$$

The distributions of the noises  $\{e_i^{d,j}\}, \{o_i^{d,j}\}$ , and  $\{o_i^f\}$  are  $B_{p_d^-}, B_{q_d^-}$ , and  $B_{\bar{p}_D}$ -bounded respectively, and  $\mathcal{Z}_{p_d}, \mathcal{Z}_{q_d}, \mathcal{Z}_{\bar{p}_D}$  are  $(\lambda^{\varepsilon^2}, \mu)$ -flawed-smudging for distributions with exactly these bounds. Note that since the number of  $d$ 's are bounded by  $D$ , which is a constant,  $\mathcal{Z}$  is the product of  $O(1)$  distributions. Hence, [Lemma 3.15](#) implies that  $\mathcal{Z}$  is  $(O(\lambda^{\varepsilon^2}), O(\mu))$ -flawed-smudging. Thus, for  $E^f, \mathcal{V}$ , there is a collection of randomized predicates  $\{\text{BAD}_\rho\}$ , one for each noise output by  $E^f$ , such that the above distributions  $\mathcal{D}_1$  and  $\mathcal{D}_2$  are identical. In addition, with probability at least  $1 - O(\mu)$ ,  $|\text{bad}|_1 = O(\lambda^{\varepsilon^2})$  and  $I$  contains at most  $O(\lambda^{\varepsilon^2})$  variables in  $(\mathbf{A}^{\leq D}, \mathbf{s}^{\leq D}, \mathbf{e}^{\leq D}, \{\mathbf{A}^d \mathbf{s}^d\})$  and at most  $O(\ell\lambda^{\varepsilon^2})$  variables in  $\mathbf{x}$ .

The fact that given the sums of noises  $E(V) + Z$ ,  $V$  is identically distributed to  $\bar{V}$ , a fresh sample that only agrees with  $V$  at locations in  $I$ , implies that in  $H_1$ , all variables can be replaced with ones computed from the fresh sample  $\bar{V}$  (while keeping the sums of noises  $E(V) + Z$  the same). This yields the following distribution of  $H_2$ :

$$H_2 = \left\{ \begin{array}{l} f \leftarrow \mathcal{FN}, V \leftarrow \mathcal{V} \\ Z \leftarrow \mathcal{Z} \\ \text{bad} \leftarrow \{\text{BAD}_\rho(E_\rho^f(V), Z)\}_\rho \\ I = \Gamma(E_{\text{bad}}^f), \bar{V} \leftarrow \mathcal{V}|_{V_I, I} \end{array} : \begin{array}{l} f, \bar{\mathbf{x}}, \\ (\bar{\mathbf{A}}^{\leq D}, \bar{\mathbf{c}}^{\leq D}), \\ \left\{ (\bar{\mathbf{A}}^{\leq D}, \bar{\mathbf{c}}_i^{\leq D}) \right\}_{i \in [t]}, \\ \text{NSim}(\text{Priv}^{f^2, \bar{\mathbf{A}}^2}, Y', \bar{\mathbf{x}}_1, \dots, \bar{\mathbf{x}}_t) \end{array} \right\},$$

where all variables with bar on top in the distribution are computed honestly from  $\bar{V}$ , and  $Y'$  is the following ‘‘hybrid’’ value, where the sums of noises are generated from  $V$ , and everything else is generated from  $\bar{V}$ :

$$\begin{aligned} Y' = & \left[ \lfloor f(\bar{\mathbf{x}}) \rfloor_{P_D} - \sum_{2 \leq d \leq D} \text{Pub}^{f^d, \bar{\mathbf{A}}^d}(\bar{\mathbf{c}}^d) \right. \\ & \left. + p_D(\mathbf{o}^f + \Phi) + \sum_{2 \leq d \leq D} \left( p_d \sum_{j \in [m]} (\mathbf{e}^{d,j} + \Phi^{d,j}) + q_d \sum_{j \in [m]} (\mathbf{o}^{d,j} + \Psi^{d,j}) \right) \right]_Q. \end{aligned}$$

The flawed smudging property guarantees that  $H_1$  and  $H_2$  are identically distributed.

**Hybrid  $H_3$**  is the same as  $H_2$ , except that we want to use the security of DHE to argue that ciphertexts  $\bar{\mathbf{c}}^{\leq D}$  encrypting  $\mathbf{x}$  in  $H_2$  can be replaced with ciphertexts encrypting zero in  $H_3$ . However, we cannot apply the security of DHE directly, as the ciphertexts are generated using variables in  $\bar{V}$ , which are not sampled completely randomly, but randomly up to  $\bar{V}_I = V_I$ . (The distribution of  $V_I$  might not be random and its value may be leaked through the sums of noises in  $Y'$ .) To circumvent this, we first single out the set  $S$  of ciphertexts  $\bar{c}_i^d = \langle \bar{\mathbf{a}}_i^d, \bar{\mathbf{s}}^d \rangle + \bar{x}_i^d + p_d \bar{e}_i^d$  such that some variable(s) in  $\bar{\mathbf{a}}_i^d$ , or the inner product  $\langle \bar{\mathbf{a}}_i^d, \bar{\mathbf{s}}^d \rangle$ , or the noise  $\bar{e}_i^d$  falls in  $\bar{V}_I = V_I$  — call them the “compromised” ciphertexts. We then argue that the secret key  $\bar{\mathbf{s}}^d$  in  $\bar{V}$  has high entropy, and thus by the robustness of HE, all ciphertexts outside  $S$ , whose  $\bar{\mathbf{a}}_i^d$ ,  $\langle \bar{\mathbf{a}}_i^d, \bar{\mathbf{s}}^d \rangle$ , and  $\bar{e}_i^d$  are sampled randomly in  $\bar{V}$ , are indistinguishable to ciphertexts encrypting zero.

More precisely, let  $S$  be the subset of ciphertexts  $\bar{c}_i^d$  such that some variable(s) in  $\bar{\mathbf{a}}_i^d$ , or  $\langle \bar{\mathbf{a}}_i^d, \bar{\mathbf{s}}^d \rangle$ , or  $\bar{e}_i^d$  are included in  $\bar{V}_I = V_I$ , or equivalently, are included in  $\text{einp}_i^{d,j}$  or  $\text{oinp}_i^{d,j}$  for some  $e_i^{d,j}$  or  $o_i^{d,j}$  in  $E_{\text{bad}}$  ( $\text{oinp}^f$  only includes  $x$  variables). By [Claim 6.9](#), if a ciphertext  $\bar{c}_i^d$  satisfies the above condition w.r.t. some  $\text{einp}_i^{d,j}$  or  $\text{oinp}_i^{d,j}$ , then it can be computed from that  $\text{einp}_i^{d,j}$  or  $\text{oinp}_i^{d,j}$ . Therefore, given  $V_I$ , all ciphertexts in  $S$  can be computed; this observation will be useful later.

We now define the hybrid  $H_3$  as

$$H_3 = \left\{ f, \bar{\mathbf{x}}, \left( \bar{\mathbf{A}}^{\leq D}, (\bar{\mathbf{c}}^{\leq D})_S, (\bar{\mathbf{c}}^{\leq D})_{\neg S} \right), \left\{ \left( \bar{\mathbf{A}}^{\leq D}, \bar{\mathbf{c}}_i^{\leq D} \right) \right\}_{i \in [t]}, \text{NSim} \left( \text{Priv}^{f^2, \bar{\mathbf{A}}^2}, Y', \bar{\mathbf{x}}_1^1, \dots, \bar{\mathbf{x}}_t^1 \right) \right\},$$

where all variables are generated identically as in  $H_2$ , except that every ciphertext outside  $S$ , say  $\bar{c}_i^d \in (\bar{\mathbf{c}}^{\leq D})_{\neg S}$ , encrypts 0, that is,  $\bar{c}_i^d = \langle \bar{\mathbf{a}}_i^d, \bar{\mathbf{s}}^d \rangle + p_d \bar{e}_i^d$ .

We now show that  $H_2$  and  $H_3$  are  $O(\mu)$ -indistinguishable for every fixed sample of  $(f, V, Z, \text{bad}, I)$  satisfying that  $|\text{bad}|_1 = O(\lambda^{\varepsilon^2})$ . This implies that  $H_2$  and  $H_3$  are  $O(\mu)$ -indistinguishable, as the condition on  $|\text{bad}|_1$  holds with probability  $1 - O(\mu)$ . We first observe that since  $V_I$  contains at most  $O(\lambda^{\varepsilon^2})$  variables in  $(\bar{\mathbf{A}}^{\leq D}, \bar{\mathbf{s}}^{\leq D}, \bar{\mathbf{e}}^{\leq D}, \{\bar{\mathbf{A}}^d \bar{\mathbf{s}}^d\})$  (and every variable contains  $O(\log n)$  bits), the min-entropy of  $\bar{\mathbf{s}}$  in  $\bar{V}$  is at least  $n - O(\lambda^{\varepsilon^2} \log n) = (1 - o(1))n$ . Next, observe that for every non-compromised ciphertext  $\bar{c}_i^d$ , none of its  $\bar{\mathbf{a}}_i^d$ , inner product  $\langle \bar{\mathbf{a}}_i^d, \bar{\mathbf{s}}^d \rangle$ , and noise  $\bar{e}_i^d$  appear in  $V_I$ , and thus  $\bar{\mathbf{a}}_i^d$  and  $\bar{e}_i^d$  are sampled randomly. Therefore, by the  $O(\mu)$ -robustness of HE w.r.t. secrets with min-entropy  $(1 - o(1))n$ , it follows that  $(\bar{\mathbf{c}}^{\leq D})_{\bar{S}}$  in  $H_3$  encrypting 0 is indistinguishable to  $(\bar{\mathbf{c}}^{\leq D})_{\bar{S}}$  in  $H_2$  encrypting values derived from  $\mathbf{x}$ .

**Hybrid  $H_4$**  is the same as  $H_3$  except that we define the simulator  $\text{Sim}$  and distribution  $\mathcal{D}_{\text{Sim}}$ , and rewrite  $H_3$  as the ideal distribution. The distribution  $\mathcal{D}_{\text{Sim}}$  samples all variables as in  $H_3$ , outputs the  $x_K$  variables that are contained in  $V_I$ , and includes all variables sampled in the state  $\text{st}$  to be passed to  $\text{Sim}$ . That is,

$$\begin{aligned} \mathcal{D}_{\text{Sim}}(f) : V &\leftarrow \mathcal{V}, Z \leftarrow \mathcal{Z}, \text{bad} \leftarrow \{\text{BAD}_\rho(E_\rho^f(V), Z)\}_\rho, I = \Gamma(E_{\text{bad}}^f), \\ &\bar{V} \leftarrow \mathcal{V}|_{V_I, I}, \text{ let } x_K \text{ be the } x \text{ variables contained in } V_I, \\ &\text{output } x_K, K, \text{st} = (V, Z, \bar{V}, \text{bad}). \end{aligned}$$

Next,  $\text{Sim}$  on input  $(\text{st}, f, y = f(\bar{\mathbf{x}}), \mathbf{x}_1, \dots, \mathbf{x}_t)$  generates variables as in  $H_3$  as follows: *i)*  $\bar{\mathbf{A}}^{\leq D}$  is included in  $\bar{V}$ ; *ii)*  $(\bar{\mathbf{c}}^{\leq D})_S$  can be computed from  $V_I$  (as observed above); *iii)*  $(\bar{\mathbf{c}}^{\leq D})_{\bar{S}}$  can be computed from  $\bar{\mathbf{A}}^{\leq D}$ ,  $\bar{\mathbf{e}}^{\leq D}$ ,  $\bar{\mathbf{s}}^{\leq D}$  in  $\bar{V}$  (encrypting 0); *iv)*  $\bar{\mathbf{x}}_i^1$  and  $\bar{\mathbf{c}}_i^{\leq D}$  depend on  $\bar{\mathbf{A}}^{\leq D}$ , and are otherwise independently sampled; *v)*  $\text{Priv}^{f^2, \mathbf{A}^2}$  depends on  $f$  and  $\bar{\mathbf{A}}^2$ ; *vi)*  $Y'$  can be computed using  $\lfloor f(\bar{\mathbf{x}}) \rfloor_{P_D}$ , all ciphertexts generated in *ii)* and *iii)*, and sums of noises generated as  $E^f(V) + Z$ . We remark that the computation of  $\text{Sim}$  depends only on  $x_K$  (in step *ii)*), and is completely independent of  $\bar{x}_{\bar{K}}$ . Therefore, given the output of  $\text{Sim}$ ,  $\bar{x}_{\bar{K}}$  is random conditioned on  $\bar{x}_K = x_K$ . Therefore,  $H_3$  is identical to the following  $\text{Ideal}$  distribution, where  $\bar{x}_{\bar{K}}$  is re-sampled again outside  $\mathcal{D}_{\text{Sim}}$ :

$$\left\{ \begin{array}{l} f \leftarrow \mathcal{FN}, \\ (x_K, K, \text{st}) \leftarrow \mathcal{D}_{\text{Sim}}(f), \quad : \quad f, \bar{\mathbf{x}}, \text{Sim}(\text{st}, f, y = f(\bar{\mathbf{x}}), \mathbf{x}_1, \dots, \mathbf{x}_t) \\ \bar{\mathbf{x}} \leftarrow \mathcal{X}|_{x_K, K}, \end{array} \right\},$$

where  $x_K$  is the set of  $x$  variables contained in  $V_I$ , which contains at most  $O(\lambda^{\varepsilon_2} \ell)$  of them.

Overall, we conclude that  $H_0$  and  $H_4$  are  $O(\mu)$ -indistinguishable, which implies that DFE has special-purpose  $O(\mu)$ -simulation security.  $\square$

### 6.3.1 Property of DFE w.r.t. a Special Purpose Function Distribution

In [Section 7](#), we will use DFE to compute functions  $f$  sampled from a special-purpose distribution  $\mathcal{FN}$  over  $\text{DF}^{N,S}$ , where the function  $f$  is determined by a fixed function  $g$  and a distributional input-output dependency graph  $G$  (sampled in  $\mathcal{FN}$ ), such that every  $f_i(\mathbf{x}, \mathbf{x}')$  equals  $g_i(x_{G(i)}, \mathbf{x}')$  and depends on a constant  $\ell = O(1)$  number of  $x$  variables specified by  $G(i)$  (the dependency on  $x'$  variables is arbitrary). In addition, the input  $\mathbf{x}$  is binary and its distribution is uniformly random (and the distribution of  $\mathbf{x}'$  is independent and arbitrary). When using DFE to compute functions on inputs from such distributions, the special-purpose simulation security guarantees that only a small set  $x_K$  of  $O(\ell \lambda^{\varepsilon_2})$   $x$  variables get compromised. We now show further that the locations  $K$  of compromised  $x$  variables only “weakly depends” on  $G$ , in the sense that there is a set  $\mathcal{K}$  independent of  $G$  such that  $G(\mathcal{K})$  contains  $K$ .

Special-purpose distributions. The function distribution  $\mathcal{FN}$  samples a function  $f = (g, G)$ , where  $g: \mathbb{Z}^N \rightarrow \mathbb{Z}^M$  is a fixed function in  $\text{DF}^{N,S}$  and  $G$  is sampled according to some distribution. For every input  $(\mathbf{x}, \mathbf{x}') \in \{0, 1\}^{N'} \times \mathbb{Z}_{p_D}^{N-N'}$  for some  $N' \leq N$  and every  $i \in [M]$ ,  $f_i(\mathbf{x}, \mathbf{x}')$  is defined to be  $g_i(x_{G(i)}, \mathbf{x}')$ . Let  $\ell$  be the locality of  $f$  on  $\mathbf{x}$ , that is,  $\ell = \max_i(|G(i)|)$ .

The input distribution  $\mathcal{X}$  on the other hand is  $(\mathbf{x}, \mathbf{x}') \leftarrow \mathcal{U}_{\{0,1\}^{N'}} \times \mathcal{X}'$ , where  $\mathbf{x}$  is a randomly and independently sampled binary string and  $\mathcal{X}'$  is arbitrary.

For any such special-purpose distributions  $\mathcal{FN}$  and  $\mathcal{X}$ , the special-purpose  $O(\mu)$ -simulation security of DFE implies the existence of an efficient and universal simulator  $\text{Sim}$  and a distribution  $\mathcal{D}_{\text{Sim}}$ , such that the  $\text{Real}$  distribution is indistinguishable to the following  $\text{Ideal}$  distribution:

$$\left\{ \begin{array}{l} f = (g, G) \leftarrow \mathcal{FN}, \\ ((x_K, x'_{K'}), (K, K'), \text{st}) \leftarrow \mathcal{D}_{\text{Sim}}(f) \quad : \quad \text{Sim}(\text{st}, f, y = f(\bar{\mathbf{x}}, \bar{\mathbf{x}}'), (\mathbf{x}_1, \mathbf{x}'_1), \dots, (\mathbf{x}_t, \mathbf{x}'_t)) \\ \bar{\mathbf{x}} \leftarrow \mathcal{U}_{\{0,1\}^{N'}}|_{x_K, K}, \bar{\mathbf{x}}' \leftarrow \mathcal{X}'|_{x'_{K'}, K'} \end{array} \right\}.$$

With probability  $1 - O(\mu)$ ,  $|K| + |K'| = O(\ell \lambda^{\varepsilon_2})$ . We now show that the locations  $K$  of compromised  $x$  variables only weakly depends on the graph  $G$ .

**Lemma 6.12.** *For every  $\lambda$ , every  $\mu$  that is superpolynomially small, and every pair of special-purpose distributions  $\mathcal{FN}$  and  $\mathcal{X}$  as described above, there exists a random variable  $\mathcal{K}$  correlated with  $(f \leftarrow \mathcal{FN}, ((x_K, x'_{K'}), (K, K'), \text{st}) \leftarrow \mathcal{D}_{\text{Sim}}(f))$ , satisfying that i)  $\mathcal{K}$  is independent of  $G$ , and ii)  $K \subseteq G(\mathcal{K})$  and  $|\mathcal{K}| = O(2^\ell \lambda^{\varepsilon^2})$  with probability  $1 - O(\sqrt{\mu})$ .*

*Proof.* Recall that  $\mathcal{D}_{\text{Sim}}$  operates as follows:

$$\begin{aligned} \mathcal{D}_{\text{Sim}}(f) : V \leftarrow \mathcal{V}, Z \leftarrow \mathcal{Z}, \text{bad} \leftarrow \{\text{BAD}_\rho(E_\rho^f(V), Z)\}_\rho, I = \Gamma(E_{\text{bad}}^f), \\ \bar{V} \leftarrow \mathcal{V}|_{V, I}, \text{ let } x_K, x'_{K'} \text{ be the } x \text{ and } x' \text{ variables contained in } V_I, \\ \text{output } (x_K, x'_{K'}), (K, K'), \text{st} = (V, Z, \bar{V}, \text{bad}). \end{aligned}$$

Further recall that each noise  $E_\rho^f(V)$  falls into one of the following three cases: It is either  $e_i^{d,j}$ ,  $o_i^{d,j}$ , or  $o_i^f$ . By Claim 6.9, for every  $f_i(\mathbf{x}) = g_i(x_{G(i)}, \mathbf{x}')$ , every  $e_i^{d,j}$ ,  $o_i^{d,j}$ , or  $o_i^f$  can be computed by i) functions  $\tilde{e}_i^{d,j}$ , or  $\tilde{o}_i^{d,j}$ , or  $\tilde{o}_i^f$  that are independent of  $G(i)$  and independent of  $G$ , on ii) inputs that depend only on  $x_{G(i)}$  and variables in  $(\mathbf{x}', \mathbf{s}^{\leq D}, \mathbf{e}^{\leq D}, \{\mathbf{A}^d \mathbf{s}^d\})$ . Overall, the function  $E^f$  is independent of  $G$  and every  $E_\rho^f(V)$  depends on  $x_{G(i)}$  for some  $i$ .

By the definition of flawed-smudging distributions, the set of randomized predicates  $\{\text{BAD}_\rho\}$  that determines the set of compromised noises depends only on  $Z$ , the function  $E$ , and the distribution  $\mathcal{V}$  (which in turn depends on the distribution of  $G$ ) of its input, all independent of the actually sampled graph  $G$ . Therefore, every  $\text{BAD}_\rho$  can be re-written as a randomized predicate  $P_\rho$  still independent of  $G$  such that

$$\text{BAD}_\rho(E_\rho(V), Z; r_\rho) = P_\rho(x_{G(i)}, \text{aux}; r_\rho),$$

for some  $i$  and  $\text{aux} = (\mathbf{x}', \mathbf{s}^{\leq D}, \mathbf{e}^{\leq D}, \{\mathbf{A}^d \mathbf{s}^d\}, Z)$ .

With probability  $1 - O(\mu)$ , the number of  $\text{BAD}_\rho$  (or equivalently the number of  $P_\rho$ ) that evaluate to 1 is bounded by  $O(\lambda^{\varepsilon^2})$ . Therefore, there is a set of  $\text{aux}$  and randomness  $r$  of all  $P_\rho$  predicates that has probability  $1 - O(\sqrt{\mu})$  of being sampled, and conditioned on them being sampled, the number of  $P_\rho(x_{G(i)}, \text{aux}; r)$  that outputs 1 is bounded by  $O(\lambda^{\varepsilon^2})$  with probability  $1 - O(\sqrt{\mu})$  over the choice of  $\mathbf{x}$  and  $G$ .

For any such  $(\text{aux}, r)$ , we have that the expectation of the sum of outputs of  $P_\rho$  is bounded:

$$\mathbb{E}_{\mathbf{x}, G} \left[ \sum_{\rho} P_\rho(\mathbf{x}_{G(i)}, \text{aux}; r) \right] = (1 - O(\sqrt{\mu})) \cdot O(\lambda^{\varepsilon^2}) + O(\sqrt{\mu}) \cdot \text{poly}(\lambda, S) = O(\lambda^{\varepsilon^2}).$$

(The first equality follows as in the rare event of  $|\text{bad}|_1$  not being bounded by  $O(\lambda^{\varepsilon^2})$ , it is still bounded by the total number of noises which is bounded by  $\text{poly}(\lambda, S)$ . Since  $\mu$  is superpolynomially small, the second equality follows.) Furthermore, let  $E_\rho = \mathbb{E}_{\mathbf{x}, G}[P_\rho(x_{G(i)}, \text{aux}; r)]$  be the expectation of  $P_\rho$  itself. Since  $|G(i)| \leq \ell$  and the marginal distribution of the binary string  $\mathbf{x}_{G(i)}$  is uniform, the expectation  $E_\rho$  is either zero if  $P_\rho(\star, \text{aux}; r)$  is constantly zero, or at least  $1/O(2^\ell)$  otherwise. By the linearity of expectation, the number of  $\rho$  s.t.  $P_\rho(\star, \text{aux}; r)$  is non-zero is at most  $O(2^\ell \lambda^{\varepsilon^2})$ . We now define the correlated random variable  $\mathcal{K}$ :

$$\mathcal{K} = \left\{ i : \exists \rho \text{ s.t. } P_\rho(\star, \text{aux}; r) \text{ is non-zero, and } P_\rho \text{ depends on } x_{G(i)} \right\}.$$

Clearly,  $|\mathcal{K}| = O(2^\ell \lambda^{\varepsilon^2})$ .

Since the above holds for a set of  $\text{aux}, r$  that appear with probability  $1 - O(\sqrt{\mu})$ , we have that with probability  $1 - O(\sqrt{\mu})$  over the choice of  $\text{aux}$  and  $r$ ,  $|\mathcal{K}| = O(2^\ell \lambda^{\varepsilon_2})$ . We further observe that  $\mathcal{K}$  is independent of  $G$ , as it only depends on the randomized predicates  $P_\rho$ , their randomness  $r$ , and  $\text{aux} = (\mathbf{x}', \mathbf{s}^{\leq D}, \mathbf{e}^{\leq D}, \{\langle \mathbf{A}^d, \mathbf{s}^d \rangle\}, Z)$ , all of which are independent of  $G$ . Finally, the set of compromised variables  $K$  must be a subset of these variables  $G(\mathcal{K})$  that non-zero predicates depend on. Therefore,  $\mathcal{K}$  is the set promised by the lemma.  $\square$

## 7 Functional Encryption for $\text{NC}^1$ and Transformation to IO

In this section, we construct sublinearly compact FE schemes for  $\text{NC}^1$  with standard fully-selective indistinguishability security, using a constant-locality PRG, the AIK randomized encoding [AIK04], our special-purpose FE scheme DFE constructed in Section 6, and a new primitive called bit-fixing homomorphic sharing. Below, we start with introducing the new primitive and constructing it from multi-key FHE in Section 7.1, and then move to the construction of FE for  $\text{NC}^1$  in Section 7.2. Finally, in Section 7.3, we describe how existing results can be used to transform our FE scheme to IO.

### 7.1 Bit-Fixing Homomorphic Sharing

A bit-fixing homomorphic sharing scheme has the same syntax as a Homomorphic Secret Sharing (HSS) scheme introduced by [BGI15, BGI16] — it enables generating a secret sharing  $x_1, \dots, x_T$  of an input  $v$ , and homomorphically evaluating a circuit  $C$  on each share separately to obtain a set of output shares  $o_1, \dots, o_T$ , from which the final output  $y = C(v)$  can be reconstructed. However, the similarity stops here, and the efficiency and security requirements are different.

- **Security:** similar to homomorphic encryption, HSS assumes that the output shares are *private*, and the shared value  $v$  is hidden when the adversary sees only a subset of  $t$  input shares. In contrast, bit-fixing homomorphic sharing requires  $v$  to be hidden even to adversaries knowing all output shares and  $t_2$  input shares. Furthermore, security needs to hold not just for honestly generated input shares, as the name suggests, but also for shares where  $t_1$  bits are fixed to arbitrary values.
- **Efficiency:** HSS (and homomorphic encryption) schemes have trivial constructions if reconstruction from the output shares can be as complex as evaluating the circuit itself (the output share can be an additive sharing of  $v$  together with  $C$ ). Therefore, HSS is only meaningful when the reconstruction is “simpler” than the computation itself. This is not the case for bit-fixing homomorphic sharing — the trivial construction of HSS does not satisfy the above security requirement — and the sharing and reconstruction procedure can take time proportional the circuit size.

We now present the formal definition:

**Definition 7.1.** A  $(T, t_1, t_2, \mu)$ -bit-fixing homomorphic sharing scheme  $\text{BF} = (\text{BFsetup}, \text{BFshare}, \text{BFeval}, \text{BFdec})$  for polynomials  $T = T(\lambda)$ ,  $t_1 = t_1(\lambda)$ , and  $t_2 = t_2(\lambda)$  consists of four efficient algorithms satisfying the following.

**Syntax:** The four algorithms have the following syntax

- $\text{BFsetup}(1^\lambda, 1^s, 1^d)$  is a randomized algorithm that, on input a security parameter  $\lambda$  and bounds  $s$  and  $d$  on the function size and depth, respectively, outputs a CRS  $\text{crs}$ .
  - $\text{BFshare}(\text{crs}, v)$  is a randomized algorithm that on input  $\text{crs}$  and  $v \in \{0, 1\}^{\text{poly}(\lambda)}$ , outputs  $T$  input shares  $x_1, \dots, x_T$  of  $v$ . Let  $n = n(\lambda)$  be the length of all input shares.
  - $\text{BFeval}(\text{crs}, x_i, i, f)$  is a deterministic algorithm that on input  $\text{crs}$ , a share  $x_i$  and its index  $i$ , and a function  $f$ , represented as a circuit of size  $s$  and depth  $d$ , outputs an output share  $o_i$ .
- Without loss of generality, if  $f$  has multiple output bits,  $\text{BFeval}$  is invoked separately for each output bit. The collection of output shares form  $o_i$ .
- $\text{BFdec}(\text{crs}, \{o_i\})$  is a deterministic algorithm that, on input  $\text{crs}$  and all output shares, outputs a string  $y$ .

**Correctness:** For every  $\lambda, s, d \in \mathbb{N}$ , every input  $v$ , and every function  $f$  of size  $s$  and depth  $d$ ,

$$\Pr \left[ \begin{array}{l} \text{crs} \leftarrow \text{BFsetup}(1^\lambda, 1^s, 1^d) \\ (x_1, \dots, x_T) \leftarrow \text{BFshare}(\text{crs}, v) \\ \forall i \in [T], o_i \leftarrow \text{BFeval}(\text{crs}, x_i, i, f) \end{array} : \text{BFdec}(\text{crs}, \{o_i\}) = f(v) \right] = 1.$$

**$(t_1, t_2, \mu)$ -bit-fixing-security** For all polynomials  $s$  and  $d$ , every sufficiently large  $\lambda \in \mathbb{N}$ ,  $s = s(\lambda)$ , and  $d = d(\lambda)$ , every pair of  $\text{poly}(\lambda)$ -bit inputs  $(v_0, v_1)$ , every  $t_1$ -bit string  $x^* \in \{0, 1\}^{t_1}$ , every set of  $t_1$  indexes  $J \subseteq [n(\lambda)]$ ,<sup>14</sup> every set of  $t_2$  indexes  $K \subseteq [T]$ , every function  $f$  of size  $s$  and depth  $d$  that does not separate  $v_0$  and  $v_1$  (i.e.,  $f(v_0) = f(v_1)$ ), and every PPT adversary  $A$ ,

$$\Pr \left[ \begin{array}{l} b \leftarrow \{0, 1\} \\ \text{crs} \leftarrow \text{BFsetup}(1^\lambda, 1^s, 1^d) \\ (x_1, \dots, x_T) \leftarrow \text{BFshare}(\text{crs}, v_b)|_{x^*, J} \\ \forall i \in [T], o_i \leftarrow \text{BFeval}(\text{crs}, x_i, i, f) \end{array} : A(\text{Binary}(x)_J, J, \{x_i\}_{i \in K}, \{o_i\}_{i \in [T]}) = b \right] \leq \frac{1}{2} + \mu(\lambda),$$

where  $\text{Binary}(x)$  denotes the binary representation of  $(x_1, \dots, x_\lambda)$  and  $\text{Binary}(x)_J$  are the bits at positions in  $J$ .

### 7.1.1 Construction from Threshold Multi-Key FHE

For any  $T, t_1$ , and  $t_2$  with  $t_1 + t_2 < T$ , a bit-fixing homomorphic sharing scheme BF scheme can be constructed from a threshold multi-key FHE scheme MFHE for functions outputting a bit, and a pseudorandom function PRF as follows:

- $\text{BFsetup}(1^\lambda, 1^s, 1^d)$  runs  $\text{params} \leftarrow \text{MFHE.Setup}(1^\lambda, 1^d)$  and outputs  $\text{crs} := \text{params}$ .
- $\text{BFshare}(\text{crs}, v)$  on input  $v \in \{0, 1\}^{\text{poly}(\lambda)}$ , computes the following:
  - $(\text{pk}_i, \text{sk}_i) \leftarrow \text{MFHE.KeyGen}(\text{crs})$  for  $i \in [T]$ ,
  - an additive sharing  $\text{ss}_1, \dots, \text{ss}_T$  of  $v$ , i.e., sample uniform bit strings  $\text{ss}_1, \dots, \text{ss}_{T-1}$  of length  $|v|$  each and set  $\text{ss}_T = v \oplus \bigoplus_{i=1}^{T-1} \text{ss}_i$ ,

<sup>14</sup>Formally, we should only quantify over  $x^*$  and  $J$  for which  $\text{BFshare}(\text{crs}, v_b)|_{x^*, J}$  is a valid distribution, i.e., for which the support of  $\text{BFshare}(\text{crs}, v_b)$  contains a bit string which coincides with  $x^*$  on the indices in  $J$ . We ignore this subtlety for ease of presentation.

- $\text{ct}_i \leftarrow \text{MFHE.Enc}(\text{pk}_i, \text{ss}_i)$  for  $i \in [T]$ ,
- $\widehat{\text{ct}}_i \leftarrow \text{MFHE.Expand}((\text{pk}_1, \dots, \text{pk}_T), i, \text{ct}_i)$  for  $i \in [T]$ .

It finally samples PRF keys  $K_i$  for  $i \in [T]$  and outputs  $(x_1, \dots, x_T)$ , where

$$x_i := \left( (\widehat{\text{ct}}_j, \text{pk}_j)_{j \in [T]}, \text{sk}_i, K_i \right).$$

- $\text{BFeval}(\text{crs}, x_i, i, f)$  on input  $\text{crs}$ ,  $x_i = ((\widehat{\text{ct}}_j, \text{pk}_j)_{j \in [T]}, \text{sk}_i, K_i)$ ,  $i$ , and a function  $f$  with  $\ell$  output bits, where  $f_j$  denotes the function computing the  $j$ th output bit, computes for all  $j \in [\ell]$ :
  - $\widehat{\text{ct}}^{(j)} = \text{MFHE.Eval}(\text{crs}, f'_j, \widehat{\text{ct}}_1, \dots, \widehat{\text{ct}}_T)$ , with  $f'_j(\text{ss}_1, \dots, \text{ss}_T) := f_j(\bigoplus_{i=1}^T \text{ss}_i) = f_j(v)$ ,
  - $r_{i,j} = \text{PRF}(K_i, j)$ ,
  - $o_{i,j} = \text{MFHE.PartDec}(\widehat{\text{ct}}^{(j)}, (\text{pk}_1, \dots, \text{pk}_T), i, \text{sk}_i; r_{i,j})$ , which means  $r_{i,j}$  is used as the randomness for the algorithm  $\text{MFHE.PartDec}$ .

It outputs the share  $o_i = (o_{i,1}, \dots, o_{i,\ell})$ .

- $\text{BFdec}(\text{crs}, \{o_i\}_{i \in [T]})$  takes as input  $\text{crs}$  and  $\{o_i\}_{i \in [T]} = \{o_{i,j}\}_{i \in [T], j \in [\ell]}$ , computes  $y_j = \text{MFHE.FinDec}(o_{1,j}, \dots, o_{T,j})$  for  $j \in [\ell]$ , and outputs  $y = (y_1, \dots, y_\ell)$ .

Correctness of the construction follows directly from the correctness of decryption of MFHE. We next show that the scheme is also bit-fixing secure.

**Theorem 7.2.** *If MFHE is  $\mu$ -semantically secure, has  $\mu$ -simulatability of partial decryptions, and PRF is  $\mu$ -pseudorandom, then BF is  $(t_1, t_2, O(L \cdot \mu))$ -bit-fixing secure for  $t_1 + t_2 < T$ , where  $L$  is an upper bound on the number of output bits of the function  $f$ .*

*Proof.* Let  $s$  and  $d$  be polynomials,  $\lambda \in \mathbb{N}$ ,  $s = s(\lambda)$ ,  $d = d(\lambda)$ , let  $v_0$  and  $v_1$  be  $\text{poly}(\lambda)$ -bit inputs, let  $x^* \in \{0, 1\}^{t_1}$ , let  $J \subseteq [n(\lambda)]$  and  $K \subseteq [T]$  with  $|J| = t_1$  and  $|K| = t_2$ , let  $f$  be a function of size  $s$  and depth  $d$  with  $\ell \leq L$  output bits and  $f(v_0) = f(v_1)$ , and let  $A$  be a PPT adversary. Now let  $H_0$  be the  $(t_1, t_2, O(L \cdot \mu))$ -bit-fixing-security experiment, i.e.,  $b \leftarrow \{0, 1\}$ ,  $\text{crs} \leftarrow \text{BFsetup}(1^\lambda, 1^s, 1^d)$ ,  $(x_1, \dots, x_T) \leftarrow \text{BFshare}(\text{crs}, v_b)|_{x^*, J}$ ,  $\forall i \in [T]$ ,  $o_i \leftarrow \text{BFeval}(\text{crs}, x_i, i, f)$ ,  $b' \leftarrow A(\text{Binary}(x)_J, J, \{x_i\}_{i \in K}, \{o_i\}_{i \in [T]})$ , and let the output of  $H_0$  be 1 if  $b' = b$ , and 0 otherwise. Note that since  $|K| = t_2$  and  $x^*$  fixes only  $t_1$  bits, where  $t_1 + t_2 < T$ , there exists an  $i_0 \in [T]$  such that no bit within  $x_{i_0}$  gets fixed and  $i_0 \notin K$ .

Define  $H_1$  to be identical to  $H_0$  except that for  $j \in [\ell]$ ,  $r_{i_0,j}$  in the execution of  $\text{BFeval}(\text{crs}, x_{i_0}, i_0, f)$  gets replaced by a truly random value. Let  $A_{\text{PRF}}$  be an adversary with access to an oracle, which is either PRF or a truly random function, and let  $A_{\text{PRF}}$  emulate the experiment toward  $A$ , where instead of running  $r_{i_0,j} = \text{PRF}(K_{i_0}, j)$ ,  $A_{\text{PRF}}$  obtains  $r_{i_0,j}$  by querying its oracle on input  $j$ . Note that this emulation can be done without  $K_{i_0}$  since no bits in  $x_{i_0}$  get fixed and  $A$  does not get  $x_{i_0}$  as an input. Finally let  $A_{\text{PRF}}$  output 1 if  $A$  guesses  $b$  correctly, and 0 otherwise. If the oracle of  $A_{\text{PRF}}$  is PRF, the view of  $A$  is identical to its view in  $H_0$ , and if the oracle is a truly random function, its view is identical to the one in  $H_1$ . Hence  $\mu$ -indistinguishability of PRF implies

$$|\Pr[H_0 = 1] - \Pr[H_1 = 1]| \leq O(\mu(\lambda)).$$

Next, we define for  $k \in \{0, \dots, \ell\}$  the hybrid  $H_{2,k}$  to be identical to  $H_1$  except that  $o_{i_0,j}$  for  $j \leq k$  is produced by the simulator  $\text{Sim}$  from  $\mu$ -simulatability of partial decryptions, where  $\text{Sim}$

is given  $i_0$ , all secret keys except  $\text{sk}_{i_0}$ ,  $\widehat{\text{ct}}^{(j)}$ , and  $f_j(v_0) = f_j(v_1)$ . Then,  $H_{2,0}$  is identical to  $H_1$  and since the statistical distance between the  $o_{i_0,j}$  in different hybrids is bounded by  $O(\mu(\lambda))$ , we have

$$\forall k \in [\ell] \quad |\Pr[H_{2,k-1} = 1] - \Pr[H_{2,k} = 1]| \leq O(\mu(\lambda)).$$

Finally let  $H_3$  be identical to  $H_{2,\ell}$  except that **BFshare** encrypts  $\perp$  instead of  $\text{ss}_{i_0}$  to obtain  $\text{ct}_{i_0}$ . Consider the attacker  $A_{\text{sem}}$  against the semantic security of **MFHE** that is given **params**, a public key  $\text{pk}^*$ , and an encryption  $\text{ct}^*$  of either  $\text{ss}_{i_0}$  or  $\perp$ , and then emulates  $H_{2,\ell}$  or  $H_3$  by using  $\text{pk}^*$  as  $\text{pk}_{i_0}$  and instead of encrypting  $\text{ss}_{i_0}$  or  $\perp$ , it uses its input  $\text{ct}^*$ . Note that this can be done without  $\text{sk}_{i_0}$  since it is only used in the scheme to compute  $\{o_{i_0,j}\}_j$  and these are simulated in both hybrids without  $\text{sk}_{i_0}$ . Finally let  $A_{\text{sem}}$  output 1 if  $A$  guesses  $b$  correctly, and 0 otherwise. We then have that the advantage of  $A_{\text{sem}}$  is bounded by  $O(\mu(\lambda))$  and thus,

$$|\Pr[H_3 = 1] - \Pr[H_{2,\ell} = 1]| \leq O(\mu(\lambda)).$$

Note that by the property of the additive sharing, the view of  $A$  in  $H_3$  is independent of  $v_b$  and thus independent of  $b$ . Hence,  $\Pr[H_3 = 1] = 1/2$ . Combined with the results above, this yields  $\Pr[H_0 = 1] \leq 1/2 + O(\ell \cdot \mu)$  and concludes the proof.  $\square$

## 7.2 Construction of FE for $\text{NC}^1$

For any fixed logarithmic function  $\text{Dep}(\lambda) = O(\log \lambda)$ , we construct a family of secret-key FE schemes  $\{\text{FE}^{N,S}\}$  for computing the class of  $\text{NC}^1$  circuits  $C$  with depth  $\text{Dep} = \text{Dep}(\lambda)$ , polynomial input length  $N = N(\lambda)$ , and size  $S = S(\lambda)$ . We will show that our schemes are sublinearly compact, and satisfy 1-key fully-selective  $\mu'$ -indistinguishability security for any  $\mu' = 2^{-o(\lambda)}$ , using the following building blocks with security levels parameterized by  $\mu = \lambda^{-\omega(1)}\mu'$ .

- A  $(\text{poly}(\lambda)S^{1-\alpha}, \text{poly}(\lambda)S)$ -stretch PRG with  $\mu$ -indistinguishability for some  $\alpha \in (0, 1/2)$  with the following structural properties. First, it has constant locality. Secondly, the function  $\text{PRG} = (P, G)$  is specified by a *single* predicate  $P: \{0, 1\}^\ell \rightarrow \{0, 1\}$  (as opposed to many predicates) and an input-output dependency graph  $G$  such that

$$\forall i \quad \text{PRG}_i(sd) = P(sd_{G(i)}).$$

Thirdly, all input nodes in the graph have degree (i.e., locality) bounded by  $o(S^{1-\alpha})$  (note that the average degree of input nodes is  $\text{poly}(\lambda)S^\alpha \ll S^{1-\alpha}$  when  $\alpha$  is a small constant). Finally, we assume PRG is a fixed function instead of a function family as in [Definition 2.8](#). Many candidate constant-locality PRGs satisfy these structural properties [[Gol00](#), [MST03](#), [OW14](#), [AL16](#)].

- A  $(T = \lambda, t_1 = \lambda/4, t_2 = \lambda/4, \mu)$ -bit-fixing homomorphic sharing scheme  $\text{BF} = (\text{BFsetup}, \text{BFshare}, \text{BFeval}, \text{BFdec})$  as constructed in the previous section.
- The AIK randomized encoding [[AIK04](#)]  $\text{RE} = (\text{REnc}, \text{REval})$  whose encoding algorithm for any function  $f$ ,  $\text{REnc}(f, x; r)$  has locality 1 in input bits and locality 3 in the random bits. The AIK randomized encodings are unconditionally and perfectly secure.
- For some fixed constant locality  $\ell$  and  $p_D = 2$ , our special-purpose FE scheme  $\text{DFE}^{N',S'} = (\text{DFE.Setup}, \text{DFE.Enc}, \text{DFE.KeyGen}, \text{DFE.Dec})$  for computing locality  $\ell$ , degree  $D = \ell$ , polynomials with input length  $N'$  and size  $S'$  over  $\mathbb{Z}_2$ , where  $N', S'$  are set below. The scheme is special-purpose  $(1 - \alpha)$ -sublinearly compact and special-purpose  $\mu^2$ -simulation secure.

We now describe our FE scheme  $\text{FE}^{N,S}$  for computing  $\text{NC}^1$  circuits with depth  $\text{Dep}$ . Below, we in-line the analysis of its correctness and  $(1 - \alpha)$ -sublinear compactness in italic font.

- $\text{FE.Setup}(1^\lambda)$ : Generate a DFE master secret key  $\text{Dmsk} \leftarrow \text{DFE.Setup}(1^\lambda, \text{pp})$ , and a CRS for the bit-fixing homomorphic sharing scheme  $\text{crs} \leftarrow \text{BFsetup}(1^\lambda, 1^s, 1^{\text{Dep}})$ , where  $s$  is described below.

Output  $\text{msk} = (\text{Dmsk}, \text{crs})$ .

- $\text{FE.KeyGen}(\text{msk}, C)$ : Input circuit  $C$  has input-length  $N$ , size  $S$ , and depth  $\text{Dep}$ . We assume for notational convenience that  $C$  has exactly  $S$  output bits, and assume w.l.o.g. that every output bit is computed by  $C_i$  with some fixed polynomial size  $s(\lambda) = \text{poly}(\lambda)$ .<sup>15</sup>

– Generate a polynomial  $f$  as follows:

- \* Divide the output bits of  $C$  into  $M = S^{1-\alpha}$  (assume for convenience that  $M$  divides  $S$ ) consecutive chunks  $I_1, \dots, I_M$ , where chunk  $I_j$  includes output bits  $(j-1)S/M + 1, \dots, jS/M$ . For every  $j \in [M]$ , let  $C_{I_j} = \{C_k\}_{k \in I_j}$  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  $C_{I_j}$ , i.e.,

$$D_i^j(x_i^j) = \text{BFeval}(\text{crs}, x_i^j, i, C_{I_j}) = \sigma_i^j.$$

Since  $\text{BFeval}$  performs homomorphic evaluation of each component  $C_i$  in  $C_{I_j}$  separately, and the component size of  $C_i$  is  $s$ , the size of each input share  $x_i^j$  is  $\text{poly}(\lambda, s)$ , and the component size of  $D_i^j$  is  $\text{poly}(\lambda, s)$  (and the overall size of  $D_i^j$  is  $\text{poly}(\lambda, s)S/M$ ).

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

$$f_i^j(x_i^j, sd) = \text{REnc}(D_i^j, x_i^j; \text{PRG}_{\Pi_i^j}(sd)).$$

Since  $\text{REnc}$  encodes the computation of every component in  $D_i^j$  separately, which takes  $\text{poly}(\lambda, s)$  time, the overall time for encoding  $D_i^j$  is  $|f_i^j| = \text{poly}(\lambda, s)S/M$ , using at most  $\phi = \text{poly}(\lambda)S/M$  random coins. Above,  $\text{PRG}_{\Pi_i^j}(sd)$  contains  $\phi$  PRG output bits at locations  $\{\pi(i, j, k)\}_{k \in [\phi]}$ , determined by the random permutation  $\pi$ . Overall,  $|\text{PRG}(sd)| = \phi \lambda M = \text{poly}(\lambda)S$  and  $|sd| = \text{poly}(\lambda)S^{1-\alpha}$ .

Finally, set

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

The input length and size of  $f$  is  $N' = |\{x_i^j\}| + |sd| = \text{poly}(\lambda, s)\lambda M + \text{poly}(\lambda)S^{1-\alpha} = \text{poly}(\lambda)S^{1-\alpha}$  and  $S' = |f_i^j|\lambda M = \text{poly}(\lambda)S$ . Since the AIK randomized encoding algorithm  $\text{REnc}$  and  $\text{PRG}$  both have constant locality,  $f$  also has constant locality  $\ell$ . Moreover, over the field  $\mathbb{Z}_2$ , it has at most degree  $\ell$ .

<sup>15</sup>If not, one can always use garbled circuits to turn  $C$  into another circuit where every output bit is computable in size  $\text{poly}(\lambda)$ , at the cost of increasing the size, input length, and output length of the circuit by a multiplicative  $\text{poly}(\lambda)$  factor.

- Generate a DFE secret key of  $f$ ,  $\text{Dsk} \leftarrow \text{DFE.KeyGen}(\text{Dmsk}, f)$ .

Output  $\text{sk} = \text{Dsk}$ .

- $\text{FE.Enc}(\text{msk}, v)$ : On input  $\text{msk} = (\text{Dmsk}, \text{crs})$  and  $v \in \{0, 1\}^N$ , do:
  - For every  $j \in [M]$ , share  $v$  using BF,  $\mathbf{x}^j = \{x_i^j\}_{i \in [\lambda]} \leftarrow \text{BFshare}(\text{crs}, v)$ .
  - Sample randomly a PRG seed  $sd \leftarrow \{0, 1\}^{\text{poly}(\lambda)S^{1-\alpha}}$ .
  - Encrypt  $X = (\{\mathbf{x}^j\}_j, sd)$  using DFE,  $\text{Dct} \leftarrow \text{DFE.Enc}(\text{Dmsk}, X)$ .

Output  $\text{ct} = \text{Dct}$ .

---

*Sublinear Compactness: It follows from the special-purpose  $(1 - \alpha)$ -sublinear compactness of DFE that  $|\text{Dct}| = \text{poly}(\lambda)(N' + S^{1-\alpha}) = \text{poly}(\lambda)S^{1-\alpha}$ . Therefore, FE is  $(1 - \alpha)$ -sublinearly compact.*

*Remark 7.3 (Special Purpose Function and Input Distributions). To show the security of FE, we need to argue that the ciphertexts of  $v^0$  and  $v^1$  are indistinguishable at the presence of a secret key for a circuit  $C$  that does not separate them. To generate a secret key for  $C$ ,  $\text{FE.KeyGen}$  generates a DFE secret key for  $f$  defined by  $C$ , and to encrypt  $v^b$  for a random  $b$ ,  $\text{FE.Enc}$  uses DFE to encrypt  $X$  defined by  $v^b$ . Consider the distributions  $\mathcal{FN}$  of  $f$  and  $\mathcal{X}$  of  $X$ . We observe that they have the form of special-purpose distributions as described in Section 6.3.1.*

*Specifically,  $\mathcal{X}$  is a product distribution  $\mathcal{X}' \times \mathcal{U}_{\{0,1\}^{\text{poly}(\lambda)S^{1-\alpha}}}$ , where the former samples the secret sharing  $\mathbf{x} = \{\mathbf{x}^j\}$  of  $v^b$  and the latter samples the PRG seed  $sd$ . Furthermore,  $f$  is defined by a fixed collection of predicates  $\{g_\rho\}$  and an input-output dependency graph  $G_\pi$  depending on a random permutation  $\pi$ . To see this, recall that every output bit  $i$  of PRG is computed using the same predicate  $P$  on a subset of seeds  $G(i)$ . Thus,  $f_i^j$  described above can be written as*

$$f_i^j(x_i^j, sd) = \text{REnc}(D_i^j, x_i^j; \text{PRG}_{\Pi_i^j}(sd) = \{P(sd_{G(\pi(i,j,k))})\}_{k \in [\phi]}).$$

*This means the collection of predicates  $\{g_\rho\}$  is fixed, defined by  $D_i^j$ ,  $P$ , and  $\text{REnc}$ , and the input-output dependency graph  $G_\pi$  is distributional, depending on the random permutation  $\pi$ , the dependency graph  $G$  of PRG, and that of  $\text{REnc}$ .*

- $\text{FE.Dec}(\text{sk}, \text{ct})$ : On input  $\text{sk} = \text{Dsk}$  and  $\text{ct} = \text{Dct}$ , do
  - Decrypt the DFE ciphertext  $\text{Dct}$  with the secret key  $\text{Dsk}$  to obtain  $y = f(X) = \text{DFE.Dec}(\text{Dsk}, \text{Dct})$ .
  - Parse  $y = \{y_i^j\}$ , and for every  $j \in [M]$  and  $i \in [\lambda]$ , decode  $y_i^j$  using  $\text{REval}$  to obtain  $\sigma_i^j = \text{REval}(y_i^j)$ .
  - For every  $j \in [M]$ , decode the output shares  $\{\sigma_i^j\}_{i \in [\lambda]}$  to obtain the actual output  $u^j = \text{BFdec}(\text{crs}, \{\sigma_i^j\})$ .

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

---

Correctness: By the correctness of DFE, we have

$$y = f(X) = \{y_i^j = f_i^j(x_i^j, sd)\}_{j,i},$$

$$y_i^j = \text{REnc}(D_i^j, x_i^j; \text{PRG}_{\Pi_i^j}(sd)).$$

By the correctness of RE, we have that

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

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

Next, we show that FE satisfies 1-key fully-selective indistinguishability-security.

**Theorem 7.4.** For all  $\mu' = 2^{-o(\lambda)}$  and for  $\mu = \lambda^{-\omega(1)}\mu'$ , assume that PRG satisfies  $\mu$ -indistinguishability, BF satisfies  $(\lambda/4, \lambda/4, \mu)$ -bit-fixing security, and  $\text{DFE}^{N', S'}$  satisfies  $\mu^2$ -special-purpose simulation security. Then,  $\text{FE}^{N, S}$  above satisfies  $\mu'$ -Sel-Ind-security.

*Proof.* Fix any  $\text{NC}^1$  circuit  $\{C\}_\lambda$  with input-length  $N$ , size  $S$ , depth  $\text{Dep}$ , and component-size  $s$ . Fix any polynomial-length sequence of pairs of inputs  $\{\{v_\rho^0, v_\rho^1\}_{\rho \in [t]}\}_\lambda$  s.t.  $C(v_\rho^0) = C(v_\rho^1)$  for every  $\rho$ . We want to show that the following distributions are  $\mu'$ -indistinguishable for  $b = 0$  and  $b = 1$ :

$$\left\{ \begin{array}{l} (\text{msk}, \text{msk}) \leftarrow \text{FE.Setup}(1^\lambda, \text{pp}) \\ \text{sk} \leftarrow \text{FE.KeyGen}(\text{msk}, C) \\ \{\text{ct}_\rho \leftarrow \text{FE.Enc}(\text{msk}, v_\rho^b)\}_{\rho \in [t]} \end{array} : \text{sk}, \{\text{ct}_\rho\}_{\rho \in [t]} \right\}.$$

To show this, it suffices to show that for every  $i^* \in [t]$ , and every PPT adversary  $A$ , and every sufficiently large  $\lambda$ , the following probability is bounded by  $1/2 + \text{poly}(\lambda)\mu$ :

$$\Pr \left[ \begin{array}{l} b \leftarrow \{0, 1\} \\ (\text{msk}, \text{msk}) \leftarrow \text{FE.Setup}(1^\lambda, \text{pp}) \\ \text{sk} \leftarrow \text{FE.KeyGen}(\text{msk}, C) \\ \text{ct} \leftarrow \text{FE.Enc}(\text{msk}, v_{i^*}^b) \\ \{\text{ct}_\rho \leftarrow \text{FE.Enc}(\text{msk}, v_\rho^1)\}_{\rho < i^*} \\ \{\text{ct}_\rho \leftarrow \text{FE.Enc}(\text{msk}, v_\rho^0)\}_{\rho > i^*} \end{array} : A(\text{sk}, \text{ct}, \{\text{ct}_\rho\}_{\rho \neq i^*}) = b \right] \leq \frac{1}{2} + \text{poly}(\lambda)\mu. \quad (14)$$

For any fixed  $i^*$  and  $A$ , we prove the above via a sequence of hybrids that gradually change the distribution of the view  $\text{view}_A = (\text{sk}, \text{ct}, \{\text{ct}_\rho\})$  of  $A$ , such that the probability that  $A$  guesses  $b$  in the last hybrid is bounded.

**Hybrid  $H_0$ :** This hybrid outputs the view  $\text{view}_A$  of  $A$  as generated in equation (14), as well as all the BF secret sharings  $\mathbf{x} = \{\mathbf{x}^j\}$  generated when encrypting  $v_{i^*}^b$ :

$$H_0 = \text{Real} = \{\mathbf{x} = \{\mathbf{x}^j\}, \text{view}_A = (\text{sk}, \text{ct}, \{\text{ct}_\rho\}_{\rho \neq i^*})\}.$$

Since every sharing  $\mathbf{x}^j$  uniquely determines  $v_{i^*}^b$ , if  $A$  is able to predict  $b$ , it means it is able to output  $b$  consistent with  $\mathbf{x}$ . By construction of FE, the secret key  $\text{sk}$  and ciphertext  $\text{ct}$

are respectively secret key and ciphertext of DFE for some function  $f$  defined by  $C$  and some input  $X$  defined by  $v_{i^*}^b$ , i.e.,

$$\text{sk} = \text{Dsk}(f), \text{ ct} = \text{Dct}(X), \text{ where } X = (\mathbf{x}, sd),$$

and similarly  $\text{ct}_\rho = \text{Dct}(X_\rho)$ , where  $X_\rho$  is defined by  $v_\rho^1$  if  $\rho < i^*$  and by  $v_\rho^0$  if  $\rho > i^*$ .

**Hybrid  $H_1$**  is the same as  $H_0$  except that the secret key  $\text{Dsk}$  and ciphertexts  $\text{Dct}, \{\text{Dct}_\rho\}$  of DFE are simulated. More precisely, let  $\mathcal{FN}$  be the distribution of  $f$ , and  $\mathcal{X}$  of  $X$  in  $H_0$ . It follows from the special-purpose  $\mu^2$ -simulation security of DFE that there exist an efficient universal simulator  $\text{Sim}$  and correlated random variables  $(X_K, K, \text{st})$  sampled by  $\mathcal{D}_{\text{Sim}}$ , such that  $H_0$  is  $\mu^2$ -indistinguishable to the following distribution:

$$H_1 = \left\{ \begin{array}{l} f \leftarrow \mathcal{FN} \\ (X_K, K, \text{st}) \leftarrow \mathcal{D}_{\text{Sim}}(f) \\ \bar{X} \leftarrow \mathcal{X}|_{X_K, K} \end{array} : \begin{array}{l} \bar{\mathbf{x}} = \{\bar{\mathbf{x}}^j\} \\ \text{view}_A = \text{Sim}\left(\text{st}, f, y = f(\bar{X}), \{\bar{X}_\rho\}_{\rho \neq i^*}\right) \end{array} \right\}.$$

Furthermore, with probability  $1 - O(\mu^2)$ ,  $|K| \leq O(\lambda^{\varepsilon_2})$  for some constant  $\varepsilon_2 \in (0, 1)$ , since  $f$  has constant locality.

Recall that the output  $y = f(\bar{X})$  consists of many randomized encodings evaluated using pseudo-random outputs of PRG,

$$y = \left\{ y_i^j = \text{REnc}(D_i^j, \bar{x}_i^j; \text{PRG}_{\Pi_i^j}(\bar{sd})) \right\}_{i,j}.$$

We would like to invoke the security of RE and PRG to argue that  $y$  can be simulated using the outputs of the encoded computations,  $\bar{\sigma}_i^j = D_i^j(\bar{x}_i^j)$ . However, this does not hold as the distribution of  $\bar{sd}$  is not completely random, but random up to agreeing with  $O(\lambda^{\varepsilon_2})$  compromised  $sd$  bits contained in  $X_K$  output by  $\mathcal{D}_{\text{Sim}}$ . Below, we single out these PRG output bits that depend on one of the compromised  $sd$  bits — call them the compromised PRG output bits as they are not guaranteed to be pseudo-random. Then, we single out these randomized encodings  $y_i^j$  that depend on one of the compromised PRG output bits — call them the compromised encodings, as they may reveal the input share  $x_i^j$  used for generating it. On the other hand, non-compromised PRG outputs are pseudo-random, and non-compromised encodings can be simulated using the corresponding output share  $\sigma_i^j$ .

More precisely, let  $sd_{K_{sd}}$  and  $\mathbf{x}_{K_x}$  denote respectively the compromised seed bits and compromised input-share bits contained in  $X_K$ . As observed in [Remark 7.3](#), the distribution  $\mathcal{X}$  of  $X$  is the product of the distribution  $\mathcal{X}'$  sampling  $\mathbf{x}$  and the uniform distribution  $\mathcal{U}$  over binary strings of length  $\{0, 1\}^{\text{poly}(\lambda)S^{1-\alpha}}$ . Thus, the bit-fixing distribution  $\mathcal{X}|_{X_K, K}$  is also a product distribution

$$\mathcal{X}|_{X_K, K} = (\bar{\mathbf{x}} \leftarrow \mathcal{X}'|_{\mathbf{x}_{K_x}, K_x}) \times (\bar{sd} \leftarrow \mathcal{U}|_{sd_{K_{sd}}, K_{sd}}).$$

In particular,  $\bar{sd}$  is uniformly random at locations outside  $K_{sd}$  (and fixed to  $sd_{K_{sd}}$  at locations in  $K_{sd}$ ). Recall that PRG has constant locality and every output bit  $\text{PRG}_i$  is computed by  $P(sd_{G(i)})$  where  $G$  is its input-output dependency graph. Let  $K_{prg}$  be the indexes of compromised PRG output bits that depend on some of the compromised seed bits in  $K_{sd}$ , that is,

$$K_{prg} = G^{-1}(K_{sd}) := \{l : G(l) \cap K_{sd} \neq \emptyset\}.$$

We next want to argue that  $\text{PRG}_{\notin K_{prg}}(\overline{sd})$  is  $\mu$ -indistinguishable to random, using that  $\overline{sd}_{\notin K_{sd}}$  is uniformly random. Note that the sampling by  $\mathcal{D}_{\text{Sim}}$  does not have to be efficient (see Definition 6.10). To use the security of PRG, we therefore need a non-uniform reduction, which is given everything not efficiently samplable as advice. (This is the reason we need PRG to be a fixed function instead of a function family from which a function is sampled, since  $f$  depends on PRG and  $\mathcal{D}_{\text{Sim}}(f)$  should not depend on the security experiment for PRG.) We then obtain  $\text{PRG}_{\notin K_{prg}}(\overline{sd}) \approx_{\mu} \mathcal{U}$ . On the other hand,  $\text{PRG}_{K_{prg}}(\overline{sd})$  is not guaranteed to be pseudorandom.

Furthermore, recall that the  $(i, j)$ 'th randomized encoding  $y_i^j$  is computed using PRG output bits at locations  $\Pi_i^j = \{\pi(i, j, k)\}_{k \in [\phi]}$ . Let  $K_{re}$  be the indexes  $(i, j)$  of compromised encodings  $y_i^j$  that depend on some compromised PRG output bits in  $\text{PRG}_{K_{prg}}(\overline{sd})$ , that is,

$$K_{re} := \{(i, j) : \Pi_i^j \cap K_{prg} \neq \emptyset\}.$$

Clearly  $K_{re}$  can be efficiently computed from  $K_{sd}$ ; for convenience, we overload the notation to also use it to denote a function  $K_{re} = K_{re}(K_{sd})$ .

All non-compromised randomized encodings  $y_i^j$  with  $(i, j) \notin K_{re}$  use pseudorandom coins. Thus, by the security of RE, again using a non-uniform reduction, non-compromised encodings can be simulated from their corresponding outputs: For every  $(i, j) \notin K_{re}$ ,

$$\left\{ y_i^j = \text{REnc} \left( D_i^j, \overline{x}_i^j ; \text{PRG}_{\Pi_i^j}(\overline{sd}) \right) \right\} \approx_{\mu} \left\{ \tilde{y}_i^j \leftarrow \text{RSim} \left( \overline{\sigma}_i^j = D_i^j(\overline{x}_i^j) \right) \right\},$$

where the output  $\overline{\sigma}_i^j$  is the output share obtained by homomorphically evaluating circuit  $C_{I_j}$  on input share  $\overline{x}_i^j$ . Therefore,  $H_1$  is  $O(\mu)$ -indistinguishable to the following hybrid  $H_2$ .

**Hybrid  $H_2$**  is the same as  $H_1$ , except that  $\text{Sim}$  simulates  $\text{view}_A$  using simulated randomized encodings at locations outside  $K_{re}$  (and still honestly generated randomized encodings at locations in  $K_{re}$ ):

$$\left\{ \begin{array}{l} f \leftarrow \mathcal{FN} \\ (X_K, K, \text{st}) \leftarrow \mathcal{D}_{\text{Sim}}(f) \\ \overline{X} \leftarrow \mathcal{X}|_{X_K, K} \end{array} : \begin{array}{l} \mathbf{x} = \{\overline{\mathbf{x}}^j\} \\ \text{view}_A = \text{Sim} \left( \text{st}, f, y' = \{y_i'^j\}_{i,j}, \{\overline{X}_\rho\}_{\rho \neq i^*} \right) \end{array} \right\},$$

where

$$\overline{X} = (\{\overline{\mathbf{x}}^j\}, \overline{sd}), \quad y_i'^j = \begin{cases} y_i^j = \text{REnc} \left( D_i^j, \overline{x}_i^j ; \text{PRG}_{\Pi_i^j}(\overline{sd}) \right) & \text{if } (i, j) \in K_{re} \\ \tilde{y}_i^j \leftarrow \text{RSim} \left( \overline{\sigma}_i^j = D_i^j(\overline{x}_i^j) \right) & \text{if } (i, j) \notin K_{re} \end{cases}.$$

We divided all the randomized encodings into  $M$  chunks – the  $j$ 'th chunk includes  $y_i^j$  for all  $i \in [\lambda]$ . These randomized encodings depend on the  $j$ 'th BF sharing  $\mathbf{x}^j = \{x_i^j\}$  of  $v_{i^*}^b$ . If there exists a  $j$  s.t. all encodings in the  $j$ 'th chunk are compromised, then the entire  $j$ 'th sharing  $\mathbf{x}^j$  is compromised, which would reveal the randomly chosen bit  $b$ . Therefore, to argue that the adversary  $A$  cannot predict  $b$ , we must argue that no such  $j$  exist. More precisely, we prove the following lemma that every chunk contains at most  $\lambda/4$  compromised encodings with very high probability — that is, compromised encodings are “*well-spread*” among different chunks.

**Lemma 7.5.** *Let  $f$  and  $K$  be sampled from  $\mathcal{FN}$  and  $\mathcal{D}_{\text{Sim}}(f)$ , respectively. We then have for all  $\mu = 2^{-o(\lambda)}$ ,*

$$\Pr[\exists j, \text{ s.t. } |\{(i, j)\}_{i \in [\lambda]} \cap K_{re}(K_{sd})| \geq \lambda/4] \leq O(\mu).$$

*Proof.* As discussed in [Remark 7.3](#), the distribution  $\mathcal{FN}$  of  $f$  and the distribution  $\mathcal{X}$  of  $X$  are special-purpose distributions as described in [Section 6.3.1](#). In particular,  $f$  is defined by a fixed function  $g$  of constant locality  $\ell$ , and a distributional input-output dependency graph  $G_\pi$  depending on a random permutation  $\pi : [\lambda] \times [M] \times [\phi]$ .

For such distributions, [Lemma 6.12](#) guarantees that there exists a correlated random variable  $\mathcal{K}$  that is independent of the graph  $G_\pi$  and hence the permutation  $\pi$ , and the set  $K_{sd}$  of compromised seed bits is a subset of  $G_\pi(\mathcal{K})$ . In addition, the size  $|\mathcal{K}|$  of  $\mathcal{K}$  is bounded by  $O(2^\ell \lambda^{\varepsilon_2})$  with probability  $1 - O(\sqrt{\mu^2}) = 1 - O(\mu)$ . Therefore, to show the lemma, it suffices to show that

$$\Pr[\exists j, \text{ s.t. } |\{(i, j)\}_{i \in [\lambda]} \cap K_{re}(G_\pi(\mathcal{K}))| \geq \lambda/4] \leq O(\mu).$$

For every single index  $\rho$ , the set  $G_\pi(\rho)$  specifies the set of seed bits that the  $\rho$ 'th output bit of  $f$  depends on. Every bit in an AIK randomized encoding depends on at most three random bits, which in turn are outputs of PRG permuted according to the random permutation  $\pi$ , thus  $G_\pi(\rho) = G(\pi(\rho_1, \rho_2, \rho_3))$ . Therefore, for the set  $\mathcal{K}$  of indexes, there exists  $\mathcal{K}'$  of size  $O(|\mathcal{K}|)$ , such that,  $G_\pi(\mathcal{K}) = G(\pi(\mathcal{K}'))$ . Recall that  $K_{re}(G(\pi(\mathcal{K}')))$  includes all indexes  $(i, j)$  of encodings  $y_i^j$  that depend on some seed bits in  $G(\pi(\mathcal{K}'))$ , more precisely,  $G(\Pi_i^j = \{\pi(i, j, k)\}_{k \in [\phi]} \cap G(\pi(\mathcal{K}')) \neq \emptyset$ . Let  $G^{-1} \circ G(i)$  represent the set of two-hop neighbors of node  $i$  in graph  $G$ . Then, the above inequality follows from the following:

$$\Pr[\exists j, \text{ s.t. } |\pi(\{(i, j, k)\}_{i \in [\lambda], k \in [\phi]}) \cap (G^{-1} \circ G(\pi(\mathcal{K}')))| \geq \lambda/4] \leq O(\mu) = 2^{-o(\lambda)}.$$

By our assumption on PRG, the input-output dependency graph  $G$  of PRG has  $n = \text{poly}(\lambda)S^{1-\alpha}$  input nodes,  $m = \text{poly}(\lambda)S$  output nodes. Furthermore, the degrees of the input nodes are bounded by  $L < o(S^{1-\alpha})$ , and the degrees of the output nodes are bounded by some constant  $l$ . The set  $\{(i, j, k)\}_{i \in [\lambda], k \in [\phi]}$  has size  $t = \text{poly}(\lambda)S^\alpha$  and  $\mathcal{K}'$  has size  $s = O(\lambda^{\varepsilon_2})$  with probability  $1 - O(\mu)$ . Importantly, both sets are independent of the random permutation  $\pi$ . Furthermore, we have  $slLt \leq m - s - t$  for sufficiently large  $S$  (relative to  $\lambda$ ) and  $\alpha < 1/2$  with probability  $1 - O(\mu)$ . Hence, the inequality follows immediately from the following claim.

**Claim 7.6.** *Let  $G$  be a bipartite graph with  $n$  input nodes and  $m$  output nodes, where the degrees of the input and output nodes are bounded by  $L$  and  $l$ , respectively. Let  $T, S \subseteq [m]$  be sets of size  $t$  and  $s$ , respectively, such that  $s \leq t$  and  $slLt \leq m - s - t$ , and let  $B \geq s + 2$ . We then have over the choice of a random permutation  $\pi$  over the output nodes (i.e.,  $\pi : [m] \rightarrow [m]$ ),*

$$\Pr_\pi [|\pi(T) \cap (G^{-1} \circ G(\pi(S)))| \geq B] \leq \exp\left(-\frac{B - s - 1}{3}\right).$$

*Proof.* Since  $\pi(S) \subseteq G^{-1} \circ G(\pi(S))$ , we have  $\pi(T \cap S) \subseteq \pi(T) \cap (G^{-1} \circ G(\pi(S)))$  for all  $\pi$ . We therefore need to bound the probability of

$$|\pi(T \setminus S) \cap (G^{-1} \circ G(\pi(S)))| \geq B - |T \cap S|.$$

The random experiment can equivalently be described as follows: First, assign to each  $j \in S$  a random, distinct  $j' \in [m]$ . Let  $S'$  be the set of all these  $j'$ , and let  $\hat{S} := G^{-1} \circ G(S')$ . Then, assign to each  $v \in T \setminus S$  a random, distinct  $v' \in [m] \setminus S'$ . Since  $|\hat{S}| \leq sL$ , the probability that the first of these  $v'$  is in  $\hat{S}$  is at most  $\frac{sL}{m-s}$ . The second  $v'$  is then chosen from a set of size  $m - s - 1$  since it must be different from the first  $v'$ . Hence, the probability that it is in  $\hat{S}$  is at most  $\frac{sL}{m-s-1}$ . Using  $|T \setminus S| \leq t$ , this implies that the probability of any  $v'$  landing in  $\hat{S}$  is at most  $\frac{sL}{m-s-t}$ . While the assignments of the  $v'$  are not independent (because they must be distinct), whether they are in  $\hat{S}$  or not only depends on previous outcomes in the sense that the probability gets smaller if previous  $v'$  are in  $\hat{S}$  (since less elements in  $\hat{S}$  are available). One can therefore define independent random variables  $X_1, \dots, X_{|T \setminus S|}$ , where each  $X_i$  is 1 with probability  $\frac{sL}{m-s-t}$ , and 0 otherwise, such that the probability of  $X_i = 1$  upper bounds the probability of the  $i$ th  $v'$  being in  $\hat{S}$ . This shows that

$$\begin{aligned} \Pr_{\pi} [|\pi(T \setminus S) \cap (G^{-1} \circ G(\pi(S)))| \geq B - |T \cap S|] \\ \leq \Pr[X_1 + \dots + X_{|T \setminus S|} \geq B - |T \cap S|] \leq \Pr[X_1 + \dots + X_{|T \setminus S|} \geq B - s]. \end{aligned}$$

Note that the expected value of  $X_1 + \dots + X_{|T \setminus S|}$  is  $\nu := \frac{sL}{m-s-t} \cdot |T \setminus S| \leq \frac{sLt}{m-s-t} \leq 1$ . Let  $\delta := \frac{B-s}{\nu} - 1$ . Since  $\nu \leq 1$  and  $B \geq s + 2$ , we have  $\delta \geq 1$ . Hence, the Chernoff bound implies

$$\Pr[X_1 + \dots + X_{|T \setminus S|} \geq B - s] = \Pr[X_1 + \dots + X_{|T \setminus S|} \geq (1 + \delta)\nu] \leq \exp(-\delta\nu/3).$$

Again using  $\nu \leq 1$ , we obtain  $\delta\nu = B - s - \nu \geq B - s - 1$ , which concludes the proof of the claim.  $\square$

This also concludes the proof of [Lemma 7.5](#).  $\square$

We now bound the probability that the PPT adversary  $A$ , on input  $\text{view}_A$ , is able to guess the random bit  $b$  underlying  $\mathbf{x} = \{\bar{\mathbf{x}}^j\}$  in  $H_2$ .

**Claim 7.7.** *For every PPT adversary  $A$ ,*

$$\Pr \left[ (\{\bar{\mathbf{x}}^j\}, \text{view}_A) \leftarrow H_2 : b \leftarrow A(\text{view}) \wedge \bar{\mathbf{x}}^1 \text{ shares } v_{i^*}^b \right] \leq 1/2 + \text{poly}(\lambda)\mu.$$

*Proof.* Fix any  $f$  and any  $(X_K, K, \text{st})$  in the support of  $\mathcal{D}_{\text{Sim}}(f)$  such that  $|K| = O(\lambda^{\varepsilon_2})$  and for all  $j$ ,  $|\{(i, j)\}_{i \in [\lambda]} \cap K_{re}\}| \leq \lambda/4$ . By [Lemma 7.5](#) and the properties of  $\mathcal{D}_{\text{Sim}}$ , such  $f$  and  $(X_K, K, \text{st})$  are sampled with probability  $1 - O(\mu)$ . Conditioned on them being sampled,  $H_2$  further samples  $\mathcal{X}|_{X_K, K}$ , which samples  $M$  sharings  $\{\bar{\mathbf{x}}^j\}$  of  $v_{i^*}^b$  for a randomly chosen bit  $b$ , conditioned on at most  $O(\lambda^{\varepsilon_2})$  compromised bits  $\mathbf{x}_{K_x}$  contained in  $X_K$ . Observe that the inputs to  $\text{Sim}$  depend only on output shares  $\{\bar{\sigma}_i^j\}_{(i,j) \notin K_{re}}$  (for simulating non-compromised encodings), and input shares  $\{\bar{x}_i^j\}_{(i,j) \in K_{re}}$  (for generating compromised encodings). Since

$|K| = O(\lambda^{\varepsilon_2})$  and  $|\{(i, j)\}_{i \in [\lambda]} \cap K_{re}\} \leq \lambda/4$  for all  $j$ , we have for every sharing  $\bar{x}^j$ , that at most  $O(\lambda^{\varepsilon_2})$  bits of it are fixed, and at most  $\lambda/4$  input shares  $\bar{x}_i^j$  are revealed to Sim. Therefore, it follows from the  $(\lambda/4, \lambda/4, \mu)$ -bit-fixing security of BF (and the fact that there are at most a polynomial number of sharings) that the probability that  $A$ , receiving the output of Sim, can predict  $b$  is at most  $1/2 + \text{poly}(\lambda)\mu$ . This concludes the proof of the claim.  $\square$

Finally, from the fact that  $H_0$  and  $H_2$  are  $O(\mu)$ -indistinguishable, we conclude that  $A$  receiving the view generated in  $H_0$  can only predict  $b$  with probability  $1/2 + \text{poly}(\lambda)\mu$ :

$$\Pr \left[ (\{\mathbf{x}^j\}, \text{view}_A) \leftarrow H_0 : b \leftarrow A(\text{view}) \wedge \mathbf{x}^1 \text{ shares } v^b \right] \leq 1/2 + \text{poly}(\lambda)\mu.$$

This concludes the proof of the theorem.  $\square$

### 7.3 From Secret-Key Ciphertext-Compact FE to IO

We describe how to apply previous works to transform our *secret-key*  $(1 - \alpha)$ -*sublinearly ciphertext-compact* FE for  $\text{NC}^1$  with subexponential (1-key) Sel-Ind-security to IO. The first FE to IO transformation was by Ananth and Jain [AJ15], and Bitansky and Vaikuntanathan [BV15], which, however, requires public-key  $(1 - \alpha)$ -sublinearly compact FE. Our FE is secret key and only satisfies a weaker notion of ciphertext-compactness, which only requires the ciphertext-size, instead of the encryption time, to be bounded by  $\text{poly}(\lambda, N)S^{1-\alpha}$ . Fortunately, following their works, Bitansky, Nishimaki, Passalegue, and Wichs [BNPW16] showed how to use sublinearly compact secret-key FE to construct IO, assuming subexponential LWE. In fact, their transformation also works for FE with only sublinear ciphertext-compactness.

In slightly more detail, they showed how to transform such FE into IO with exponential efficiency, called XIO [LPST16b], whose obfuscated circuits have size  $\text{poly}(|C|, \lambda)2^{(1-\beta)n}$  where  $|C|$  and  $n$  are respectively the size and input-length of the original circuit and  $\beta$  is an arbitrary constant in  $(0, 1)$  — that is, the size of the obfuscated circuit is sublinear in the size of the truth table of the original circuit.

**Theorem 7.8** (Following from Remark 3.1 in [BNPW16] (also see Construction 2 in Section 7.2 in the full version of [BLP17])). *If there exists sublinearly ciphertext-compact FE for  $\text{NC}^1$  with subexponential Sel-Ind-security, then there exists subexponentially secure XIO for  $\text{NC}^1$ .*

Lin, Pass, Seth, and Telang [LPST16b] showed that, despite of having exponentially large obfuscated circuits, XIO implies fully-efficient IO, assuming subexponential LWE.

**Theorem 7.9** ([LPST16b]). *Assume subexponential security of the LWE assumption. If there exists subexponentially secure XIO for  $\text{NC}^1$ , then there exists subexponentially secure IO for P/poly.*

Combining our sublinearly ciphertext-compact FE schemes for  $\text{NC}^1$  with the above two theorems gives IO for P/poly.

## 8 Weakening Requirements on Flawed-Smudging Distributions

In this section, we show how flawed-smudging distributions in our constructions can be replaced by something weaker, namely distributions  $\mathcal{X}$  that are flawed-smudging only with some small

probability  $\varepsilon$ . This can be formalized as follows: there exists an event  $\text{bad}$  with probability  $1 - \varepsilon$  such that  $X$  conditioned on  $\neg\text{bad}$  is flawed-smudging. For our construction, we only need  $\varepsilon = 1/\text{poly}(\lambda)$ . As we argue below, using such distributions in our construction yields an FE scheme that is secure with probability  $\varepsilon' = \text{poly}(\varepsilon)$ . We then show how to amplify the security of that FE scheme. The basic idea is to use a bit-fixing homomorphic sharing to share the input and encrypt each share using an independent FE scheme. We use a sharing scheme that tolerates leakage of all but one share. Thus, since each FE scheme is secure with probability  $1/\text{poly}(\lambda)$ , polynomially many shares are sufficient to guarantee that at least one of them remains secure.

## 8.1 Flawed-Smudging Distributions with Small Probability

We first give a formal definition of a distribution that is flawed-smudging with probability  $\varepsilon$ .

**Definition 8.1.** Let  $\ell$  be a positive integer and let  $\mathcal{X}$  and  $\mathcal{E}$  be distributions over  $\mathbb{Z}^\ell$ . Further let  $K \in \mathbb{N}$  and let  $\mu, \varepsilon \in [0, 1]$ . We say that  $\mathcal{X}$  is  $(K, \mu)$ -flawed-smudging for  $\mathcal{E}$  with probability  $\varepsilon$  if there exists an event  $\text{bad}$  such that  $\Pr[\text{bad}] = 1 - \varepsilon$  and the conditional distribution  $\mathcal{X}|\neg\text{bad}$  is  $(K, \mu)$ -flawed-smudging for  $\mathcal{E}$ .

*Remark 8.2.* The definition above implies that if a distribution  $\mathcal{X}$  is flawed-smudging with probability  $\varepsilon$ , then  $\mathcal{X}$  is  $(1 - \varepsilon)$ -indistinguishable from a flawed-smudging distribution, namely  $\mathcal{X}' := \mathcal{X}|\neg\text{bad}$ : For any distinguisher  $D$ , and for  $X \leftarrow \mathcal{X}$ , and  $X' \leftarrow \mathcal{X}'$ ,

$$\begin{aligned} & |\Pr[D(X) = 1] - \Pr[D(X') = 1]| \\ &= |\Pr[\text{bad}] \cdot \Pr[D(X) = 1 \mid \text{bad}] + \Pr[\neg\text{bad}] \cdot \Pr[D(X) = 1 \mid \neg\text{bad}] - \Pr[D(X') = 1]| \\ &= |\Pr[\text{bad}] \cdot \Pr[D(X) = 1 \mid \text{bad}] + (1 - \Pr[\text{bad}]) \cdot \Pr[D(X') = 1] - \Pr[D(X') = 1]| \\ &\leq \Pr[\text{bad}] = 1 - \varepsilon. \end{aligned}$$

The reverse implication, however, does not necessarily hold, i.e., assuming  $\mathcal{X}$  is flawed-smudging with probability  $\varepsilon$  is a stronger assumption than assuming that  $\mathcal{X}$  is  $(1 - \varepsilon)$ -indistinguishable from a flawed-smudging distribution. The reason is that the event  $\text{bad}$  in [Definition 8.1](#) depends only on  $X$ , but not on the random coins of the distinguisher  $D$ . In contrast to that,  $\Delta$ RGs by Ananth et al. [[AJKS18](#)] are defined via computational indistinguishability between distributions. Therefore, our amplification in [Section 8.3](#) is simpler than the corresponding amplification in [[AJKS18](#)], which needs to amplify computational security.

## 8.2 Using Weaker Distributions in Previous Constructions

We now describe how the weaker distributions from [Definition 8.1](#) can be used in our previous constructions. We start with the construction of FE for constant degree polynomials in [Section 6.3](#). This construction uses an  $\eta$ -noisy secret-key FE scheme NFE, where  $\eta$  is of the form

$$p_D \Phi + \sum_{2 \leq d \leq D} \left( p_d \cdot \sum_{j \in [m]} \Phi^{d,j} + q_d \cdot \sum_{j \in [m]} \Psi^{d,j} \right),$$

and  $\Phi$ ,  $\Phi^{d,1} \parallel \dots \parallel \Phi^{d,m}$ , and  $\Psi^{d,1} \parallel \dots \parallel \Psi^{d,m}$  are sampled from the  $(\lambda^{\varepsilon^2}, \mu)$ -flawed-smudging distributions  $\mathcal{Z}_{\bar{p}_D}$ ,  $\mathcal{Z}_{p_d}$ , and  $\mathcal{Z}_{q_d}$ , respectively. We now assume that  $\mathcal{Z}_{\bar{p}_D}$ ,  $\{\mathcal{Z}_{p_d}\}_{2 \leq d \leq D}$ , and  $\{\mathcal{Z}_{q_d}\}_{2 \leq d \leq D}$  are only  $(\lambda^{\varepsilon^2}, \mu)$ -flawed-smudging with probability  $\varepsilon = 1/\text{poly}(\lambda)$  each, and denote the resulting distribution by  $\tilde{\eta}$ , and the corresponding NFE scheme by  $\tilde{\text{NFE}}$ . Then, there exist

events  $\text{bad}_\star$  for  $\star \in \{\bar{p}_D, p_d, q_d \mid 2 \leq d \leq D\}$  such that  $\Pr[\text{bad}_\star] = 1 - \varepsilon$  and  $\mathcal{Z}_\star \mid \neg \text{bad}_\star$  is  $(\lambda^{\varepsilon_2}, \mu)$ -flawed-smudging. We define the event

$$\text{bad}' := \text{bad}_{\bar{p}_D} \cup \bigcup_{2 \leq d \leq D} (\text{bad}_{p_d} \cup \text{bad}_{q_d}).$$

We then have that conditioned on the event  $\neg \text{bad}'$ , all the distributions above are flawed-smudging, and since  $D$  is a constant,

$$\varepsilon' := \Pr[\neg \text{bad}'] = \varepsilon^{2D-1} = 1/\text{poly}(\lambda).$$

Let  $\widetilde{\text{DFE}}$  be the scheme constructed when NFE in the construction in [Section 6.3](#) is replaced by  $\widetilde{\text{NFE}}$ . Note that the proof of special-purpose simulation security of the scheme DFE (see [Lemma 6.11](#)) only uses the flawed-smudging property of the distributions in one  $\eta$  from a single NFE ciphertext (in Hybrid  $H_2$ ). Hence, the overall security of the scheme  $\widetilde{\text{DFE}}$  holds conditioned on the event  $\neg \text{bad}'$  for the corresponding distribution  $\tilde{\eta}$ . We summarize this in the following lemma (cf. [Lemma 6.11](#)).

**Lemma 8.3.** *Assume that DHE has  $\mu$ -robustness for secrets with  $(1 - o(1))n$  min-entropy, for all polynomials  $N'$  and  $S$ ,  $\widetilde{\text{NFE}}^{N', S}$  satisfies 1-key  $O(\mu)$ -Sel-Sim security. Then, for all polynomials  $N$  and  $S$ , for every distribution  $\{\mathcal{FN}\}_\lambda$  over  $f \in \text{DF}^{N, S}$ , every distribution  $\{\mathcal{X}\}_\lambda$  over  $\mathbf{x}$  in  $\mathbb{Z}_{p_D}^N$ , and every sequence of vectors  $\{\mathbf{x}_1, \dots, \mathbf{x}_t\}$ , where  $\mathbf{x}_i \in \mathbb{Z}_{q_D}^N$ , there exists a (potentially inefficient)  $\mathcal{D}_{\text{Sim}}$  sampling  $(x_K, K, \text{st})$ , such that the following distributions are  $O(\mu)$ -indistinguishable:*

$$\left\{ \begin{array}{l} f \leftarrow \mathcal{FN}, \mathbf{x} \leftarrow \mathcal{X} \\ \text{msk} \leftarrow \widetilde{\text{DFE}}.\text{Setup}(1^\lambda) \\ \text{sk} \leftarrow \widetilde{\text{DFE}}.\text{KeyGen}(\text{msk}, f) \\ \text{ct} \leftarrow \widetilde{\text{DFE}}.\text{Enc}(\text{msk}, \mathbf{x}) \\ \{\text{ct}_i \leftarrow \widetilde{\text{DFE}}.\text{Enc}(\text{msk}, \mathbf{x}_i)\}_{i \in [t]} \end{array} : f, \mathbf{x}, (\text{sk}, \text{ct}, \text{ct}_1, \dots, \text{ct}_t) \mid \neg \text{bad}' \right\}_{\lambda \in \mathbb{N}},$$

$$\left\{ \begin{array}{l} f \leftarrow \mathcal{FN}, \\ (x_K, K, \text{st}) \leftarrow \mathcal{D}_{\text{Sim}}(f), \\ \bar{\mathbf{x}} \leftarrow \mathcal{X}|_{x_K, K}, \end{array} : f, \bar{\mathbf{x}}, \text{Sim}(\text{st}, f, y = f(\bar{\mathbf{x}}, \mathbf{x}_1, \dots, \mathbf{x}_t)) \right\}_{\lambda \in \mathbb{N}},$$

where  $\text{bad}'$  is the event corresponding to the distribution  $\tilde{\eta}$  of the  $\widetilde{\text{NFE}}$  ciphertext in  $\text{ct}$  with  $\Pr[\neg \text{bad}'] = \varepsilon'$ , and with probability at least  $1 - \mu$ ,  $|K| \leq O(\lambda^{\varepsilon_2} \ell)$  for some constant  $\varepsilon_2 \in (0, 1)$ , where  $\ell$  is the maximum locality of  $f \in \text{DF}^{N, S}$ .

We refer to this as  $O(\mu)$ -security with probability  $\varepsilon'$  for  $\widetilde{\text{DFE}}$ .

Next, we consider the construction of FE for  $\text{NC}^1$  in [Section 7.2](#). The construction of FE uses DFE, which we now replace by  $\widetilde{\text{DFE}}$  to obtain the scheme  $\widetilde{\text{FE}}$ . In [Theorem 7.4](#),  $\mu'$ -Sel-Ind-security of FE is proven by first proving [equation \(14\)](#), which only requires indistinguishability of a single pair of FE ciphertexts, which each correspond to a single DFE ciphertext. The security of DFE is then used only once to show that hybrids  $H_0$  and  $H_1$  are indistinguishable. Using [Lemma 8.3](#), we can thus conclude that  $H_0$  for  $\widetilde{\text{FE}}$  conditioned on  $\neg \text{bad}'$  is indistinguishable from  $H_1$ . As in the proof of [Theorem 7.4](#),  $H_1$  and  $H_2$  are indistinguishable and in  $H_2$ , the adversary's advantage to guess the bit  $b$  can be bounded. Hence, the equivalent of [equation \(14\)](#) for  $\widetilde{\text{FE}}$  holds when conditioning on  $\neg \text{bad}'$ . Note, however, that one cannot obtain  $\mu'$ -Sel-Ind-security of  $\widetilde{\text{FE}}$ , which

requires indistinguishability of polynomially many ciphertext pairs, since that would require polynomially many bad events to not occur. The following lemma summarizes our observation (cf. [Theorem 7.4](#)).

**Lemma 8.4.** *For  $\mu = 2^{-o(\lambda)}$ , assume that PRG satisfies  $\mu$ -indistinguishability, BF satisfies  $(\lambda/4, \lambda/4, \mu)$ -bit-fixing security, and  $\widetilde{\text{DFE}}^{N', S'}$  satisfies  $\mu^2$ -security with probability  $\varepsilon'$  from [Lemma 8.3](#). Let  $\{C\}_\lambda$  be  $\text{NC}^1$  circuits with input-length  $N$ , size  $S$ , depth  $\text{Dep}$ , and component-size  $s$ , let  $\{v_\rho\}_{\rho \in [t], \lambda \in \mathbb{N}}$  be a polynomial-length sequence of inputs, and let  $\{v_0^*, v_1^*\}_\lambda$  be inputs such that  $C(v_0^*) = C(v_1^*)$ . Then, for every PPT adversary  $A$  and for all sufficiently large  $\lambda$ ,*

$$\Pr \left[ \begin{array}{l} b \leftarrow \{0, 1\} \\ (\text{msk}, \text{msk}) \leftarrow \widetilde{\text{FE}}.\text{Setup}(1^\lambda, \text{pp}) \\ \text{sk} \leftarrow \widetilde{\text{FE}}.\text{KeyGen}(\text{msk}, C) \\ \text{ct}^* \leftarrow \widetilde{\text{FE}}.\text{Enc}(\text{msk}, v_b^*) \\ \{\text{ct}_\rho \leftarrow \widetilde{\text{FE}}.\text{Enc}(\text{msk}, v_\rho)\}_{\rho \in [t]} \end{array} \quad : \quad A(\text{sk}, \text{ct}^*, \{\text{ct}_\rho\}_{\rho \in [t]}) = b \quad \neg \text{bad}' \right] \leq \frac{1}{2} + \text{poly}(\lambda)\mu ,$$

where  $\text{bad}'$  is the event for the distribution in the ciphertext  $\text{ct}^*$ .

We refer to this as  $\text{poly}(\lambda)\mu$ -security with probability  $\varepsilon'$  for  $\widetilde{\text{FE}}$ .

### 8.3 Amplifying Security

We finally show how to construct a  $\mu'$ -Sel-Ind-secure scheme FE from  $\widetilde{\text{FE}}$ . As mentioned before, the main idea of this construction is to share the inputs with a bit-fixing homomorphic sharing scheme and to encrypt each share with an independent instance of  $\widetilde{\text{FE}}$ . A technical issue we have to deal with is that [Lemma 8.4](#) only guarantees indistinguishability security for  $\widetilde{\text{FE}}$  instead of simulation security. We use the technique from [\[DCIJ<sup>+</sup>13, ABSV15\]](#) to resolve this as follows: Ciphertexts encrypt a flag that is always 0 in an honest execution. We generate keys for functions that check this flag and evaluate the regular function if the flag is 0, but otherwise decrypt a ciphertext of a symmetric encryption scheme that is embedded in the function. This allows us to embed the actual function output into the secret keys and then use indistinguishability security to change the ciphertexts to encrypting the zero-string with the flag set to 1.

For any fixed logarithmic function  $\text{Dep}(\lambda) = O(\log \lambda)$ , we construct a family of secret-key FE schemes  $\{\widetilde{\text{FE}}^{N, S}\}$  for computing the class of  $\text{NC}^1$  circuits  $C$  with depth  $\text{Dep} = \text{Dep}(\lambda)$ , polynomial input length  $N = N(\lambda)$ , and size  $S = S(\lambda)$  using the following building blocks.

- The scheme  $\widetilde{\text{FE}}^{N', S'} = (\widetilde{\text{FE}}.\text{Setup}, \widetilde{\text{FE}}.\text{Enc}, \widetilde{\text{FE}}.\text{KeyGen}, \widetilde{\text{FE}}.\text{Dec})$  described above for computing locality  $\ell$ , degree  $D = \ell$ , polynomials with input length  $N'$  and size  $S'$  over  $\mathbb{Z}_2$ , where  $N'$  and  $S'$  are set below.
- A  $(T = \log(\mu)/\log(1 - \varepsilon'), t_1 = 0, t_2 = T - 1, \mu)$ -bit-fixing homomorphic sharing scheme  $\text{BF} = (\text{BFsetup}, \text{BFshare}, \text{BFeval}, \text{BFdec})$ , that has CRS and input shares with sizes independent of the size of supported functions. Note that since  $\text{BFsetup}$  takes the size and depth of the functions as input, these values can in general depend on the function size. In our construction in [Section 7.1.1](#), however, they only depend on the depth and not on the size of the functions. Furthermore, we require for any  $f$  corresponding to a circuit of size  $S$  that  $\text{BFeval}(\text{crs}, \cdot, i, f)$  is a circuit of size  $\text{poly}(\lambda)S$ . This is also the case for our construction in [Section 7.1.1](#) when instantiated with the MFHE scheme from [\[MW16\]](#).

Note that since  $\lim_{n \rightarrow \infty} (1 - 1/n)^n = 1/e$ , we have  $\lim_{n \rightarrow \infty} n \log(1 - 1/n) = -1$ . Hence,  $n$  and  $-1/\log(1 - 1/n)$  have the same asymptotic behavior. This implies that  $-1/\log(1 - \varepsilon')$  is  $\text{poly}(\lambda)$  since  $\varepsilon' = 1/\text{poly}(\lambda)$ . Therefore,  $T = \log(\mu)/\log(1 - \varepsilon')$  is also polynomial in  $\lambda$ .

- A symmetric encryption scheme  $\text{Sym}$  with  $\mu$ -IND-CPA security and deterministic decryption algorithm in  $\text{NC}^1$ .

**Construction of FE.** Our scheme FE consists of the following algorithms:

- $\text{FE.Setup}(1^\lambda)$  on input a security parameter  $1^\lambda$ , generates  $T$   $\widetilde{\text{FE}}$  master secret keys  $\widetilde{\text{msk}}_i \leftarrow \widetilde{\text{FE.Setup}}(1^\lambda)$  and symmetric encryption keys  $K_i \leftarrow \text{Sym.KeyGen}(1^\lambda)$  for  $i \in [T]$ , and a CRS for the bit-fixing homomorphic sharing scheme  $\text{crs} \leftarrow \text{BFsetup}(1^\lambda, 1^S, 1^{\text{Dep}})$ .

It outputs  $\text{msk} = (\{\widetilde{\text{msk}}_i, K_i\}_{i \in [T]}, \text{crs})$ .

- $\text{FE.Enc}(\text{msk}, v)$  on input  $\text{msk} = (\{\widetilde{\text{msk}}_i, K_i\}_{i \in [T]}, \text{crs})$  and  $v \in \{0, 1\}^N$ , shares  $v$  using BF,  $\{x_i\}_{i \in [T]} \leftarrow \text{BFshare}(\text{crs}, v)$ . Then, for every  $i \in [T]$ , it encrypts  $x_i$ ,  $0^{|K_i|}$ , and  $\text{flag} := 0$  using  $\widetilde{\text{FE}}$ :

$$\text{ct}_i \leftarrow \widetilde{\text{FE.Enc}}(\widetilde{\text{msk}}_i, (x_i, 0^{|K_i|}, \text{flag} = 0)).$$

It finally outputs  $\text{ct} = \{\text{ct}_i\}_{i \in [T]}$ .

- $\text{FE.KeyGen}(\text{msk}, f)$  on input  $\text{msk} = (\{\widetilde{\text{msk}}_i, K_i\}_{i \in [T]}, \text{crs})$  and a function  $f$  represented as a circuit with input-length  $N$ , size  $S$ , and depth  $\text{Dep}$ , does for  $i \in [T]$ :

- Let  $g_{f,i}$  be the function that on input the  $i$ 'th share  $x_i$  of the sharing  $\mathbf{x}$  of  $v$ , homomorphically evaluates  $f$ , i.e.,

$$g_{f,i}(x_i) = \text{BFeval}(\text{crs}, x_i, i, f) = o_i.$$

- Compute  $c_i \leftarrow \text{Sym.Enc}(K_i, 0^{|o_i|})$ , where  $|o_i|$  is the length of an output share.
- Let  $h_{f,i,c_i}$  be the function that on input  $(x_i, K_i, \text{flag})$  evaluates  $y = g_{f,i}(x_i)$  if  $\text{flag} = 0$ , and if  $\text{flag} = 1$ , computes  $y = \text{Sym.Dec}(K_i, c_i)$ :

$$h_{f,i,c_i}(x_i, K_i, \text{flag}) := (1 - \text{flag}) \cdot g_{f,i}(x_i) + \text{flag} \cdot \text{Sym.Dec}(K_i, c_i).$$

- Generate a  $\widetilde{\text{FE}}$  secret key for  $h_{f,i,c_i}$ ,  $\widetilde{\text{sk}}_i \leftarrow \widetilde{\text{FE.KeyGen}}(\widetilde{\text{msk}}_i, h_{f,i,c_i})$ .

Output  $\text{sk} = \{\widetilde{\text{sk}}_i\}_{i \in [T]}$ .

Since the sizes of input shares of BF do not depend on the sizes of the supported functions, the length of an input to  $h_{f,i,c_i}$  is  $N' = |x_i| + \text{poly}(\lambda) = \text{poly}(\lambda, N)$ . Furthermore, by our assumption on  $\text{BFeval}$ , the size of  $\text{BFeval}(\text{crs}, \cdot, i, f)$  is  $\text{poly}(\lambda)S$ . Hence, the size of  $h_{f,i,c_i}$  is also  $S' = \text{poly}(\lambda)S$ .

- $\text{FE.Dec}(\text{sk}, \text{ct})$  on input  $\text{sk} = \{\widetilde{\text{sk}}_i\}_{i \in [T]}$  and  $\text{ct} = \{\text{ct}_i\}_{i \in [T]}$ , computes  $o_i \leftarrow \widetilde{\text{FE.Dec}}(\widetilde{\text{sk}}_i, \text{ct}_i)$  for  $i \in [T]$ , and then outputs  $y \leftarrow \text{BFdec}(\text{crs}, \{o_i\}_{i \in [T]})$ .

**Correctness.** Let  $f$  be a function represented as a circuit with input-length  $N$ , size  $S$ , and depth  $\text{Dep}$ , and let  $x$  be an input. Further let  $\text{msk} \leftarrow \text{FE.Setup}(1^\lambda)$ ,  $\text{ct} \leftarrow \text{FE.Enc}(\text{msk}, x)$ ,  $\text{sk} \leftarrow \text{FE.KeyGen}(\text{msk}, f)$ , and  $y \leftarrow \text{FE.Dec}(\text{sk}, \text{ct})$ . We then have by correctness of  $\widetilde{\text{FE}}$  that  $o_i$  computed by  $\text{FE.Dec}$  equals

$$o_i = h_{f,i,c_i}(x_i, 0^{|K_i|}, \text{flag} = 0) = \text{BFeval}(\text{crs}, x_i, i, f),$$

for  $\{x_i\}_{i \in [T]} \leftarrow \text{BFshare}(\text{crs}, v)$ . Since  $\text{FE.Dec}$  outputs  $y \leftarrow \text{BFdec}(\text{crs}, \{o_i\}_{i \in [T]})$ , the correctness of  $\text{BF}$  implies that  $y = f(v)$ .

**Sublinear compactness.** The size of a ciphertext  $\text{ct} = \{\text{ct}_i\}_{i \in [T]}$  is  $T = \text{poly}(\lambda)$  times the size of an  $\widetilde{\text{FE}}$  ciphertext  $\text{ct}_i$ . The  $(1 - \alpha)$ -sublinearly compactness of  $\widetilde{\text{FE}}$  implies that  $\text{ct}_i$  has size  $\text{poly}(\lambda, N') \cdot S'^{1-\alpha} = \text{poly}(\lambda, N) \cdot S^{1-\alpha}$ . Thus,  $\text{FE}$  is  $(1 - \alpha)$ -sublinearly compact.

**Single-key, fully selective indistinguishability security.** We finally prove that  $\text{FE}$  is  $\mu'$ -Sel-Ind-secure.

**Theorem 8.5.** *For all  $\mu' = 2^{-o(\lambda)}$  and for  $\mu = \lambda^{-\omega(1)}\mu'$ , assume that  $\text{Sym}$  is  $\mu$ -IND-CPA secure,  $\text{BF}$  satisfies  $(T = \log(\mu)/\log(1 - \epsilon'), t_1 = 0, t_2 = T - 1, \mu)$ -bit-fixing security, and  $\widetilde{\text{FE}}^{N', S'}$  satisfies  $\text{poly}(\lambda)\mu$ -security with probability  $\epsilon'$  from Lemma 8.4. Then,  $\text{FE}^{N, S}$  satisfies  $\mu'$ -Sel-Ind-security.*

*Proof.* Let  $\{f\}_\lambda$  be functions with input-length  $N$ , size  $S$ , and depth  $\text{Dep}$ , and let  $\{\{v_\rho^0, v_\rho^1\}_{\rho \in [t]}\}_\lambda$  be a polynomially long sequence of inputs such that  $f(v_\rho^0) = f(v_\rho^1)$  for every  $\rho$ . As in the proof of Theorem 7.4, it suffices to show that for every  $i^* \in [t]$ , every PPT adversary  $A$ , and every sufficiently large  $\lambda$ , the following probability is bounded by  $1/2 + \text{poly}(\lambda)\mu$ :

$$\Pr \left[ \begin{array}{l} b \leftarrow \{0, 1\} \\ (\text{msk}, \text{msk}) \leftarrow \text{FE.Setup}(1^\lambda) \\ \text{sk} \leftarrow \text{FE.KeyGen}(\text{msk}, f) \\ \text{ct}^* \leftarrow \text{FE.Enc}(\text{msk}, v_{i^*}^b) \\ \{\text{ct}_\rho \leftarrow \text{FE.Enc}(\text{msk}, v_\rho^1)\}_{\rho < i^*} \\ \{\text{ct}_\rho \leftarrow \text{FE.Enc}(\text{msk}, v_\rho^0)\}_{\rho > i^*} \end{array} : A(\text{sk}, \text{ct}^*, \{\text{ct}_\rho\}_{\rho \neq i^*}) = b \right] \leq \frac{1}{2} + \text{poly}(\lambda)\mu .$$

For any fixed  $i^*$  and  $A$ , we prove the above via a sequence of hybrids changing the view  $\text{view}_A = (\text{sk}, \text{ct}^*, \{\text{ct}_\rho\})$  of  $A$ , such that the probability that  $A$  guesses  $b$  in the last hybrid is bounded.

**Hybrid  $H_0$ :** In this hybrid,  $\text{view}_A$  is the real distribution with all values generated honestly. In particular, we have  $\text{sk} = \{\tilde{\text{sk}}_i\}_{i \in [T]}$  with  $\tilde{\text{sk}}_i \leftarrow \widetilde{\text{FE.KeyGen}}(\widetilde{\text{msk}}_i, h_{f,i,c_i})$ ,

$$h_{f,i,c_i}(x_i, K_i, \text{flag}) = (1 - \text{flag}) \cdot \text{BFeval}(\text{crs}, x_i, i, f) + \text{flag} \cdot \text{Sym.Dec}(K_i, c_i),$$

and  $c_i \leftarrow \text{Sym.Enc}(K_i, 0^{|o_i|})$ . Furthermore,  $\text{ct}^* = \{\text{ct}_i^*\}_{i \in [T]}$  with

$$\text{ct}_i^* \leftarrow \widetilde{\text{FE.Enc}}(\widetilde{\text{msk}}_i, (x_i^*, 0^{|K_i|}, \text{flag} = 0)),$$

and  $\{x_i^*\}_{i \in [T]} \leftarrow \text{BFshare}(\text{crs}, v_{i^*}^b)$ .

**Hybrid  $H_1$**  is the same as  $H_0$  except that we replace the symmetric ciphertexts  $c_i$  for  $i \in [T]$  with encryptions of  $o_i = \text{BFeval}(\text{crs}, x_i^*, i, f)$ :

$$c_i \leftarrow \text{Sym.Enc}(K_i, o_i).$$

Note that  $K_i$  is only used to generate  $c_i$ , which is never decrypted. We can therefore apply the security of  $\text{Sym}$ . Since we change  $T = \text{poly}(\lambda)$  ciphertexts,  $\mu$ -IND-CPA-security of  $\text{Sym}$  implies that the hybrids  $H_0$  and  $H_1$  are  $\text{poly}(\lambda)\mu$ -indistinguishable.

**Hybrid  $H_2$ :** By the security of  $\widetilde{\text{FE}}$ , there are independent events  $\text{bad}'_1, \dots, \text{bad}'_T$  such that conditioned on  $\neg \text{bad}'_i$ , the ciphertext  $\text{ct}_i^*$  is  $\text{poly}(\lambda)\mu$ -indistinguishable from an encryption of a different value that evaluates to the same as the value encrypted in  $\text{ct}_i^*$  under  $h_{f,i,c_i}$ . We define  $H_2$  to be identical to  $H_1$ , but it additionally samples  $\text{bad}'_1, \dots, \text{bad}'_T$  and for every  $i$  with  $\neg \text{bad}'_i$ , it replaces  $\text{ct}_i^*$  by

$$\text{ct}_i^* \leftarrow \widetilde{\text{FE}}.\text{Enc}(\widetilde{\text{msk}}_i, (0^{|x_i^*|}, K_i, \text{flag} = 1)).$$

We have by construction  $h_{f,i,c_i}(0^{|x_i^*|}, K_i, \text{flag} = 1) = o_i = h_{f,i,c_i}(x_i^*, 0^{|K_i|}, \text{flag} = 0)$ . Hence, we can use the  $\text{poly}(\lambda)\mu$ -security with probability  $\varepsilon'$  from [Lemma 8.4](#) to conclude that  $H_1$  and  $H_2$  are  $\text{poly}(\lambda)\mu$ -indistinguishable.

We finally bound the advantage of  $A$  guessing  $b$  in  $H_2$ . Note that the view of  $A$  in  $H_2$  only depends on  $o_i$  and is otherwise independent of  $x_i^*$  for all  $i$  with  $\neg \text{bad}'_i$ . Thus, if  $\neg \text{bad}'_i$  occurs for at least one  $i$ , we can use the security of  $\text{BF}$  to conclude that  $A$  can guess  $b$  with probability at most  $1/2 + \mu$ . For each  $i$ ,  $\Pr[\text{bad}'_i] = 1 - \varepsilon'$ . Hence, the probability that  $\text{bad}'_i$  occurs for all  $i \in [T]$  is

$$\Pr \left[ \bigwedge_{i \in [T]} \text{bad}'_i \right] = (1 - \varepsilon')^T = (1 - \varepsilon')^{\frac{\log \mu}{\log(1 - \varepsilon')}} = \mu.$$

Therefore, the probability that  $A$  guesses  $b$  correctly in  $H_2$  is at most  $1/2 + 2\mu$ . Note that sampling the events  $\text{bad}'_i$  in Hybrid  $H_2$  is not necessarily efficient. Hence, to reduce to the security of  $\text{BF}$ , we need a non-uniform reduction that receives the  $\text{bad}'_i$  together with the distributions  $\tilde{\eta}_i$  as advice.

Overall, we can conclude that  $A$  cannot guess  $b$  with probability better than  $1/2 + \text{poly}(\lambda)\mu$ .  $\square$

**Acknowledgments.** Many thanks to Shweta Agrawal for sharing an early version of [\[Agr18a\]](#), and to Stefano Tessaro for many general discussions.

The authors are supported by NSF grants CNS-1528178, CNS-1514526, CNS-1652849 (CA-REER), a Hellman Fellowship, the Defense Advanced Research Projects Agency (DARPA) and Army Research Office (ARO) under Contract No. W911NF-15-C-0236, and a subcontract No. 2017-002 through Galois. The views expressed are those of the authors and do not reflect the official policy or position of the Department of Defense, the National Science Foundation, or the U.S. Government.

## 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.

- [ABSV15] Prabhanjan Ananth, Zvika Brakerski, Gil Segev, and Vinod Vaikuntanathan. From selective to adaptive security in functional encryption. In Rosario Gennaro and Matthew Robshaw, editors, *Advances in Cryptology – CRYPTO 2015*, pages 657–677, Berlin, Heidelberg, 2015. Springer.
- [ADGM17] Daniel Apon, Nico Döttling, Sanjam Garg, and Pratyay Mukherjee. Cryptanalysis of indistinguishability obfuscations of circuits over GGH13. In Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl, editors, *44th International Colloquium on Automata, Languages, and Programming (ICALP 2017)*, volume 80 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 38:1–38:16, Dagstuhl, Germany, 2017. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.
- [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.
- [AGIS14] 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.
- [Agr17] 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.
- [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.
- [AIK06] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Computationally private randomizing polynomials and their applications. *computational complexity*, 15(2):115–162, 6 2006.
- [AJ15] Prabhanjan Ananth and Abhishek Jain. Indistinguishability obfuscation from compact functional encryption. In Rosario Gennaro and Matthew Robshaw, editors, *Advances in Cryptology – CRYPTO 2015*, pages 308–326, Berlin, Heidelberg, 2015. Springer.
- [AJKS18] Prabhanjan Ananth, Aayush Jain, Dakshita Khurana, 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 Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *Advances in Cryptology – EUROCRYPT 2017*, pages 152–181, Cham, 2017. Springer International Publishing.
- [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.
- [BGI<sup>+</sup>01] Boaz Barak, Oded Goldreich, Rusell Impagliazzo, Steven Rudich, Amit Sahai, Salil Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In Joe Kilian, editor, *Advances in Cryptology – CRYPTO 2001*, pages 1–18, Berlin, Heidelberg, 2001. Springer.
- [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.
- [BGI16] Elette Boyle, Niv Gilboa, and Yuval Ishai. Breaking the circuit size barrier for secure computation under ddh. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology – CRYPTO 2016*, pages 509–539, Berlin, Heidelberg, 2016. Springer.
- [BGK<sup>+</sup>14] Boaz Barak, Sanjam Garg, Yael Tauman Kalai, Omer Paneth, and Amit Sahai. Protecting obfuscation against algebraic attacks. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology – EUROCRYPT 2014*, pages 221–238, Berlin, Heidelberg, 2014. Springer.
- [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.

- [BHK<sup>+</sup>18] Boaz Barak, Sam Hopkins, Pravesh Kothari, Aayush Jain, and Amit Sahai. Talk at TCS workshop at Crypto, 2018.
- [BLP17] Nir Bitansky, Huijia Lin, and Omer Paneth. On removing graded encodings from functional encryption. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *Advances in Cryptology – EUROCRYPT 2017*, pages 3–29, Cham, 2017. Springer International Publishing.
- [BNPW16] Nir Bitansky, Ryo Nishimaki, Alain Passelègue, and Daniel Wichs. From cryptomania to obfustopia through secret-key functional encryption. In Martin Hirt and Adam Smith, editors, *Theory of Cryptography*, pages 391–418, Berlin, Heidelberg, 2016. Springer.
- [BQ12] Andrej Bogdanov and Youming Qiao. On the security of Goldreich’s one-way function. *Comput. Complex.*, 21(1):83–127, March 2012.
- [BR14] 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.
- [BS03] Dan Boneh and Alice Silverberg. Applications of multilinear forms to cryptography. *Contemporary Mathematics*, 324(1):71–90, 2003.
- [BSW11] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In Yuval Ishai, editor, *Theory of Cryptography (TCC)*, pages 253–273, Berlin, Heidelberg, 2011. Springer.
- [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.
- [BV15] Nir Bitansky and Vinod Vaikuntanathan. Indistinguishability obfuscation from functional encryption. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 171–190, 10 2015.
- [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<sup>+</sup>15] Jean-Sébastien Coron, Craig Gentry, Shai Halevi, Tancreède 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 Rosario Gennaro and Matthew Robshaw, editors, *Advances in Cryptology – CRYPTO 2015*, pages 247–266, Berlin, Heidelberg, 2015. Springer.
- [CGH17] Yilei Chen, Craig Gentry, and Shai Halevi. Cryptanalyses of candidate branching program obfuscators. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *Advances in Cryptology – EUROCRYPT 2017*, pages 278–307, Cham, 2017. Springer International Publishing.

- [CHL<sup>+</sup>15] Jung Hee Cheon, Kyoohyung Han, Changmin Lee, Hansol Ryu, and Damien Stehlé. Cryptanalysis of the multilinear map over the integers. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015*, pages 3–12, Berlin, Heidelberg, 2015. Springer.
- [CLT13] Jean-Sébastien Coron, Tancrede 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.
- [CLT15] Jean-Sébastien Coron, Tancrede 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.
- [DCIJ<sup>+</sup>13] Angelo De Caro, Vincenzo Iovino, Abhishek Jain, Adam O’Neill, Omer Paneth, and Giuseppe Persiano. On the achievability of simulation-based security for functional encryption. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013*, pages 519–535, Berlin, Heidelberg, 2013. Springer.
- [DGG<sup>+</sup>16] 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>.
- [DORS08] Yevgeniy Dodis, Rafail Ostrovsky, Leonid Reyzin, and Adam Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *SIAM Journal on Computing*, 38(1):97–139, 2008.
- [Gen09a] Craig Gentry. *A fully homomorphic encryption scheme*. PhD thesis, Stanford University, 2009. [crypto.stanford.edu/craig](http://crypto.stanford.edu/craig).
- [Gen09b] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the Forty-first Annual ACM Symposium on Theory of Computing*, STOC ’09, pages 169–178, New York, NY, USA, 2009. ACM.
- [GGH<sup>+</sup>13] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, pages 40–49, 10 2013.
- [GGH15] Craig Gentry, Sergey Gorbunov, and Shai Halevi. Graph-induced multilinear maps from lattices. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *Theory of Cryptography*, pages 498–527, Berlin, Heidelberg, 2015. Springer.
- [GKP<sup>+</sup>13] Shafi Goldwasser, Yael Kalai, Raluca Ada Popa, Vinod Vaikuntanathan, and Nikolai Zeldovich. Reusable garbled circuits and succinct functional encryption. In *Proceedings of the Forty-fifth Annual ACM Symposium on Theory of Computing*, STOC ’13, pages 555–564, New York, NY, USA, 2013. ACM.

- [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.
- [GLSW15] Craig Gentry, Allison Bishop Lewko, Amit Sahai, and Brent Waters. Indistinguishability obfuscation from the multilinear subgroup elimination assumption. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*, pages 151–170, 10 2015.
- [GMM<sup>+</sup>16] Sanjam Garg, Eric Miles, Pratyay Mukherjee, Amit Sahai, Akshayaram Srinivasan, and Mark Zhandry. Secure obfuscation in a weak multilinear map model. In Martin Hirt and Adam Smith, editors, *Theory of Cryptography*, pages 241–268, Berlin, Heidelberg, 2016. Springer.
- [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.
- [GVW12] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Functional encryption with bounded collusions via multi-party computation. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology – CRYPTO 2012*, pages 162–179, Berlin, Heidelberg, 2012. Springer.
- [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.
- [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.
- [KNT18] Fuyuki Kitagawa, Ryo Nishimaki, and Keisuke Tanaka. Obustopia built on secret-key functional encryption. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018*, pages 603–648, Cham, 2018. Springer International Publishing.
- [LATV12] Adriana López-Alt, Eran Tromer, and Vinod Vaikuntanathan. On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In *Proceedings of the Forty-fourth Annual ACM Symposium on Theory of Computing, STOC '12*, pages 1219–1234, New York, NY, USA, 2012. ACM.
- [Lin16] 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 Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology – CRYPTO 2017*, pages 599–629, Cham, 2017. Springer International Publishing.
- [LPST16a] Huijia Lin, Rafael Pass, Karn Seth, and Sidharth Telang. Indistinguishability obfuscation with non-trivial efficiency. In Chen-Mou Cheng, Kai-Min Chung, Giuseppe Persiano, and Bo-Yin Yang, editors, *Public-Key Cryptography – PKC 2016*, pages 447–462, Berlin, Heidelberg, 2016. Springer.
- [LPST16b] Huijia Lin, Rafael Pass, Karn Seth, and Sidharth Telang. Output-compressing randomized encodings and applications. In Eyal Kushilevitz and Tal Malkin, editors, *Theory of Cryptography*, pages 96–124, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg.
- [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 *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 11–20, 10 2016.
- [LV17] Alex Lombardi and Vinod Vaikuntanathan. Limits on the locality of pseudorandom generators and applications to indistinguishability obfuscation. In Yael Kalai and Leonid Reyzin, editors, *Theory of Cryptography*, pages 119–137, Cham, 2017. Springer International Publishing.
- [MPR07] Ueli Maurer, Krzysztof Pietrzak, and Renato Renner. Indistinguishability amplification. In Alfred Menezes, editor, *Advances in Cryptology - CRYPTO 2007*, pages 130–149, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.
- [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 Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology – CRYPTO 2016*, pages 629–658, Berlin, Heidelberg, 2016. Springer.
- [MW16] Pratyay Mukherjee and Daniel Wichs. Two round multiparty computation via multi-key FHE. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology – EUROCRYPT 2016*, pages 735–763, Berlin, Heidelberg, 2016. Springer.

- [O’N10] Adam O’Neill. Definitional issues in functional encryption. Cryptology ePrint Archive, Report 2010/556, 2010. <https://eprint.iacr.org/2010/556>.
- [OW14] Ryan O’Donnell and David Witmer. Goldreich’s PRG: Evidence for near-optimal polynomial stretch. In *2014 IEEE 29th Conference on Computational Complexity (CCC)*, pages 1–12, June 2014.
- [Pei09] Chris Peikert. Public-key cryptosystems from the worst-case shortest vector problem: Extended abstract. In *Proceedings of the Forty-first Annual ACM Symposium on Theory of Computing, STOC ’09*, pages 333–342, New York, NY, USA, 2009. ACM.
- [PST14] Rafael Pass, Karn Seth, and Sidharth Telang. Indistinguishability obfuscation 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 *Proceedings of the Thirty-seventh Annual ACM Symposium on Theory of Computing, STOC ’05*, pages 84–93, New York, NY, USA, 2005. ACM.
- [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.