Indistinguishability Obfuscation from Simple-to-State Hard Problems: New Assumptions, New Techniques, and Simplification

Romain Gay[1], Aayush Jain[2], Huijia Lin[3], and Amit Sahai[2]

[1] IBM, Zurich, Switzerland
romain.rgay@gmail.com,
[2] UCLA, Los Angeles, CA 90095, USA
aayushjain@cs.ucla.edu
sahai@cs.ucla.edu,
[3] University of Washington, Seattle, WA 98195, USA
rachel@cs.washington.edu

Abstract. In this work, we study the question of what set of simple-to-state assumptions suffice for constructing functional encryption and indistinguishability obfuscation ($i\mathcal{O}$), supporting all functions describable by polynomial-size circuits. Our work improves over the state-of-the-art work of Jain, Lin, Matt, and Sahai (Eurocrypt 2019) in multiple dimensions.

New Assumption: Previous to our work, all constructions of $i\mathcal{O}$ from simple assumptions required novel pseudorandomness generators involving LWE samples and constant-degree polynomials over the integers, evaluated on the error of the LWE samples. In contrast, Boolean pseudorandom generators (PRGs) computable by constant-degree polynomials have been extensively studied since the work of Goldreich (2000).[4] We show how to replace the novel pseudorandom objects over the integers used in previous works, with appropriate Boolean pseudorandom generators with sufficient stretch, when combined with LWE with binary error over suitable parameters. Both binary error LWE and constant degree Goldreich PRGs have been a subject of extensive cryptanalysis since much before our work and thus we back the plausibility of our assumption with security against algorithms studied in context of cryptanalysis of these objects.

New Techniques: we introduce a number of new techniques:

- We show how to build partially-hiding *public-key* functional encryption, supporting degree-2 functions in the secret part of the message, and arithmetic NC^1 functions over the public part of the message, assuming only standard assumptions over asymmetric pairing groups.
- We construct single-ciphertext secret-key functional encryption for all circuits with *linear* key generation, assuming only the LWE assumption.

[4] Goldreich and follow-up works study Boolean pseudorandom generators with constant-locality, which can be computed by constant-degree polynomials.

SIMPLIFICATION: Unlike prior works, our new techniques furthermore let us construct *public-key* functional encryption for polynomial-sized circuits directly (without invoking any bootstrapping theorem, nor transformation from secret-key to public key FE), and based only on the *polynomial hardness* of underlying assumptions. The functional encryption scheme satisfies a strong notion of efficiency where the size of the ciphertext grows only sublinearly in the output size of the circuit and not its size. Finally, assuming that the underlying assumptions are subexponentially hard, we can bootstrap this construction to achieve $i\mathcal{O}$.

1 Introduction

This paper studies the notion of indistinguishability obfuscation ($i\mathcal{O}$) for general programs computable in polynomial time [22, 51, 41], and develops several new techniques to strengthen the foundations of $i\mathcal{O}$. The key security property for $i\mathcal{O}$ requires that for any two equivalent programs P_0 and P_1 modeled as circuits of the same size, where "equivalent" means that $\mathsf{P}_0(x) = \mathsf{P}_1(x)$ for all inputs x, we have that $i\mathcal{O}(\mathsf{P}_0)$ is computationally indistinguishable to $i\mathcal{O}(\mathsf{P}_1)$. Furthermore, the obfuscator $i\mathcal{O}$ should run in probabilistically polynomial time.

This notion of obfuscation was coined by [22] in 2001. However, until 2013, there was not even a single candidate construction known. This changed with the breakthrough work of [41]. Soon after, the floodgates opened and a flurry of over 100 papers were published reporting applications of $i\mathcal{O}$ (e.g. [76, 34, 49, 56, 60, 25] [39, 43, 55]). Not only did $i\mathcal{O}$ enable the first constructions of numerous important cryptographic primitives, $i\mathcal{O}$ also *expanded* the scope of cryptography, allowing us to mathematically approach problems that were previously considered the domain of software engineering. A simple example along these lines is the notion of *crippleware* [41]: Alice, a software developer, has developed a program P using powerful secrets, and wishes to sell her work. Before requiring payment, Alice is willing to share with Bob a weakened (or "crippled") version of her software. Now, Alice could spend weeks developing this crippled version $\widetilde{\mathsf{P}}$ of her software, being careful not to use her secrets in doing so; or she could simply disable certain inputs to cripple it yielding an equivalent P', but this would run the risk of Bob hacking her software to re-enable those disabled features. $i\mathcal{O}$ brings this problem of software engineering into the realm of mathematical analysis. With $i\mathcal{O}$, Alice could avoid weeks of effort by simply giving to Bob $i\mathcal{O}(\mathsf{P}')$, and because this is indistinguishable from $i\mathcal{O}(\widetilde{\mathsf{P}})$, Alice is assured that Bob can learn no secrets.

Not only has $i\mathcal{O}$ been instrumental in realizing new cryptographic applications, it has helped us advance our understanding of long-standing theoretical questions. One such recent example is that of the first cryptographic evidence of the average-case hardness of the complexity class PPAD (which contains of the problem of finding Nash equilibrium). In particular, [25] constructed hard instances for the End Of the Line (EOL) problem assuming subexponentially secure $i\mathcal{O}$ and one-way functions.

Our Contributions. In this work, we show how to simplify, both technically and conceptually, the task of constructing secure $i\mathcal{O}$ schemes. Notably, the ideas we develop in this work helped pave the way for the recent first construction of $i\mathcal{O}$ from well-studied assumptions [58], resolving the central open question in the area of $i\mathcal{O}$. The follow-up work of [58] builds upon this paper.

We now discuss the contributions of our paper in detail.

What hardness assumptions suffice for constructing $i\mathcal{O}$? Given its importance, a crucial question is to identify what hardness assumptions, in particular, simple ones, suffice for constructing $i\mathcal{O}$. While it is hard to concretely measure simplicity in assumptions, important features include i) having succinct description, ii) being falsifiable and instance independent (e.g., independent of the circuit being obfuscated), and iii) consisting of only a constant number of assumptions, as opposed to families of an exponential number of assumptions. However, research on this question has followed a tortuous path over the past several years, and so far, despite of a lot of progress, before our work, no known $i\mathcal{O}$ constructions [41, 21, 32, 10, 19, 42, 38, 62, 14, 47, 72, 66, 63, 65, 46, 23, 3, 5, 11, 57, 30] were based on assumptions that have all above features.

Our new assumption. In this work, building upon assumptions introduced in [11, 57], we introduce a new simple-to-state assumption, that satisfies all the features enumerated above. We show how to provably achieve $i\mathcal{O}$ based only on our new assumption combined with standard assumptions, namely subexponentially secure Learning With Errors (LWE) problem [73], and subexponentially secure SXDH and bilateral DLIN assumptions over bilinear maps [59, 28]. Let us now describe, informally, our new assumption. In this introductory description, we will omit discussion of parameter choices; however, they are crucial (even for standard assumptions), and we discuss them in detail in our technical sections. We start by describing the ingredients that will go into the assumption.

Constant-degree[5] Boolean PRGs generalize constant-locality Boolean PRGs, as for Boolean functions, locality upper bounds the degree. The latter is tightly connected to the fundamental topic of Constraint Satisfaction Problems (CSPs) in complexity theory, and were first proposed for cryptographic use by Goldreich [48] 20 years ago. The complexity theory and cryptography communities have jointly developed a rich body of literature on the cryptanalysis and theory of constant-locality Boolean PRGs [48, 69, 16, 27, 15, 70, 17, 40]. Our new assumption first postulates that there exists a constant d-degree Boolean PRG, $G : \{0,1\}^n \to \{0,1\}^m$ with sufficient stretch $m \geq n^{\lceil \frac{d}{2} \rceil \cdot (0.5+\epsilon)+\rho}$ for some constants $\epsilon, \rho > 0$, whose output $\boldsymbol{r} = G(\boldsymbol{x})$ should satisfy the standard notion of pseudorandomness. Furthermore, our assumption postulates that the pseudorandomness holds even when its Boolean input $\boldsymbol{x} \in \{0,1\}^n$ is embedded in LWE samples as noises, and the samples are made public. The latter is known as *Learning With Binary Errors (LWBE)*, which has been studied over the

[5] throughout this work, unless specified, by degree of boolean PRGs, we mean the degree of the polynomial computing the PRG over the reals.

last decade [68, 18, 36, 37]. Our new assumption, combining Boolean PRGs and LWBE, is as follows:

The G-LWEleak*-security assumption (informal).*

$$\left(\{\boldsymbol{a}_i, \langle \boldsymbol{a}_i, \boldsymbol{s}\rangle + e_i \bmod p\}_{i\in[n]}, G, G(\boldsymbol{e})\right) \; //\boldsymbol{e} = (e_1, \ldots, e_n) \leftarrow \{0,1\}^n,\ \boldsymbol{a}_i, \boldsymbol{s} \leftarrow \mathbb{Z}_p^{n^{0.5+\epsilon}}$$

$$\approx \quad \left(\{\boldsymbol{a}_i, \langle \boldsymbol{a}_i, \boldsymbol{s}\rangle + e_i \bmod p\}_{i\in[n]}, G, \boldsymbol{r}\right) \; //\boldsymbol{r} \leftarrow \{0,1\}^m$$

As is evident here, this assumption is quite succinct, is falsifiable and instance-independent, does not involve an exponential family of assumptions, and does not use multilinear maps. Furthermore, the ingredients that make up the assumption – Constant-degree Boolean PRGs and LWBE – have a long history of study within cryptography and complexity theory. As we discuss in detail in the full version, this assumption avoids attacks by all known cryptanalytic techniques. We note that the parameter n of LWBE samples is chosen to be sub-quadratic in the length $|\boldsymbol{s}|$ of the secret. This is needed in order to avoid Arora-Ge attacks on LWBE [18], and also avoid all known algebraic attacks [36]. Indeed, the parameter choices we make are not possible using the previous work of [57], and the parameters used in [57] would render LWBE insecure.

Comparison of our assumption with the subsequent follow-up work of [58]. Our shift to considering Boolean PRGs in the context of the approach of [57] provided a conceptual starting point for the subsequent work of [58], which finally achieved $i\mathcal{O}$ from four well-founded assumptions: LPN over $\mathbb{F}_p$, LWE, Boolean PRGs in NC^0, and SXDH. Indeed, the work of [58] essentially succeeds in "separating" the two ingredients in our assumption above — that is, basing $i\mathcal{O}$ on LWBE and the security of Goldreich's PRG with appropriate parameters separately, through a novel leveraging of the LPN over $\mathbb{F}_p$ assumption. Indeed, their work goes further and actually eliminates the need for the LWBE assumption entirely, and also eliminates the parameter requirements that we needed for Goldreich's PRG.

Complexity and clarity in $i\mathcal{O}$ constructions. Another motivation for our work is to address the complexity of existing $i\mathcal{O}$ constructions. Current constructions of $i\mathcal{O}$ are rather complex in the sense they often rely on many intermediate steps, each of which incur a complexity blow up, both in the sense of computational complexity and in the sense of difficulty of understanding. Ideally, for the sake of simplicity, $i\mathcal{O}$ schemes would minimize the number of such transformations, and instead aim at a more direct construction. In our case, we solely rely on the generic transformation of [13, 26], which shows that $i\mathcal{O}$ can be build from Functional Encryption [77], a primitive that was originally formulated by [29, 71]. Roughly speaking, FE is a public-key or secret-key encryption scheme where users can generate restricted decryption keys, called functional keys, where each such key is associated with a particular function f. Such a key allows the decryptor to learn from an encryption of a plaintext m, the value $f(m)$, and nothing beyond that.

Previous constructions fell short in directly constructing a full-fledged FE needed for the implication of $i\mathcal{O}$ [13, 26]. For example, the work of [57] first obtain a "weak" FE that: i) is *secret-key*, ii) only generates function keys associated with function computable *only by* NC_0 *circuits*, iii) only ensures *weak security*, and iv) is based on subexponential hardness assumptions. Then, generic transformations are applied to "lift" the function class supported and the security level, which inevitably makes the final FE and $i\mathcal{O}$ schemes quite complex.

This state of affairs motivates simplifying $i\mathcal{O}$ constructions, for efficiency and simplicity itself, but also for making a technically deep topic more broadly accessible to the community. That is also one of the goals of this paper.

1.1 Our Results

Our main result is a simpler and more direct $i\mathcal{O}$ construction from the following assumptions.

Theorem 1. *There is a construction of $i\mathcal{O}$ for obfuscating all polynomial-sized circuits based on the following assumptions:*

- *There exists a constant-degree d Boolean PRG $G : \{0,1\}^n \to \{0,1\}^m$ with sufficient stretch $m \geq n^{\lceil \frac{d}{2} \rceil \cdot (0.5+\epsilon)+\rho}$ for some constant $\epsilon, \rho > 0$, and satisfies subexponential* $\mathsf{G\text{-}LWEleak}$*-security,*
- *the subexponential LWBE assumption, and*
- *the subexponential bilateral DLIN and SXDH assumption over asymmetric pairing groups.*

Our techniques and additional results. Our construction of FE and $i\mathcal{O}$ are enabled by our new assumption and a number of new techniques designed to enable basing the security of $i\mathcal{O}$ on simple-to-state assumptions. We briefly summarize them here, but we elaborate on how they are used in the $i\mathcal{O}$ construction in the technical overview section immediately following this introduction.

Single-Ciphertext Functional Encryption with Linear Key Generation. We construct, assuming only LWE, a single-ciphertext secret-key functional encryption scheme able to give functional keys associated with any polynomial-sized circuit with depth bounded by λ, whose key generation and decryption algorithms have certain *simple structures*: i) The key generation algorithm computes a *linear* function on the master secret key and randomness, and ii) the decryption algorithm, given a ciphertext ct, a functional secret key sk_f associated with a function f and the description of f itself, first performs some deterministic computation on the ciphertext to get an intermediate ciphertext ct_f, followed by simply subtracting the sk_f from it, and then rounds to obtain the outcome. This object is previously known as special homomorphic encryption in the literature [6, 3, 64]. However, prior constructions only handles functional keys associated with NC_0 circuits (for those based on LWE) or NC^1 circuits (for those based on ring LWE). In this work, we view it through the FE lens, and construct it from LWE for all functions computable by polynomial-size circuits with any depth bounded by

the security parameter λ. Constructing such single-ciphertext (or single-key) FE (that do not have compact ciphertexts) from standard assumptions is a meaningful goal on its own. In the literature, there are constructions of single-ciphertext FE from the minimal assumption of public-key encryption [74, 52], and several applications (e.g., [9]). However, they do not have the type of simple structures (e.g., linear key generation algorithm) our construction enjoys, and consequently cannot be used in our $i\mathcal{O}$ construction. These simple structural properties may also find uses in other applications.

Partially-Hiding Functional Encryption for NC^1 *Public Computation and Degree-2 Private Computation.* Partially-hiding Functional Encryption (PHFE) schemes involve functional secret keys, each of which is associated with some 2-ary function f, and decryption of a ciphertext encrypting $(\boldsymbol{x}, \boldsymbol{y})$ with such a key reveals $f(\boldsymbol{x}, \boldsymbol{y})$, $\boldsymbol{x}$, f, and nothing more about $\boldsymbol{y}$. Since only the input $\boldsymbol{y}$ is hidden, such an FE scheme is called partially-hiding FE. The notion was originally introduced by [53] where it was used to bootstrap FE schemes. A similar notion of partially-hiding predicate encryption was proposed and constructed by [54]. PHFE beyond the case of predicate encryption was first constructed by [12] for functions f that compute degree-2 polynomials on the input $\boldsymbol{y}$ and degree-1 polynomials in $\boldsymbol{x}$, under the name of 3-restricted FE, in the secret-key setting. In this work, we construct a PHFE scheme from standard assumptions over bilinear pairing groups, that is *public-key* and supports functions f that have degree 2 in the private input $\boldsymbol{y}$, while performs an arithmetic NC^1 computation on the public input $\boldsymbol{x}$, More precisely, $f(\boldsymbol{x}, \boldsymbol{y}) = \langle g(\boldsymbol{x}), q(\boldsymbol{y}) \rangle$ where g is computable by an arithmetic log-depth circuit and q is a degree-2 polynomial. The previous best constructions of partially-hiding FE were secret-key, and could only handle NC_0 computation on the public input [57].

This contribution is interesting in its own right, as a step forward towards broadening the class of functions supported by FE schemes from standard assumptions. In particular, it can be used to combine rich access-control and perform selective computation on the encrypted data. In that context, the public input $\boldsymbol{x}$ represents some attributes, while the private input $\boldsymbol{y}$ is the plaintext. Functional secret keys reveal the evaluation of a degree-2 polynomial on the private input if some policy access, represented by an NC^1 arithmetic circuit evaluates to true on the attributes. This is the key-policy variant of a class of FE with rich access-control introduced in [2]. In the latter, the authors build an FE scheme where ciphertexts encrypt a Boolean formula (the public input) and a vector (the private input). Functional secret keys are associated with attributes and a vector of weights, and decryption yields the weighted sum of the plaintexts if the formula embedded in the ciphertext evaluates to true on the attributes embedded in the functional secret key. Their construction, as ours, rely on standard pairing assumptions, but only permits computation of *degree-1* polynomials on the private input. They also give a lattice-based construction, which is limited to identity-based access structures.

2 Technical Overview

Below, we will use several different encryption schemes, and adopt the following notation to refer to ciphertexts and keys of different schemes. For a scheme x (e.g., a homomorphic encryption scheme HE, or a functional encryption scheme FE), we denote by $\mathsf{xct}, \mathsf{xsk}$ a ciphertext, or secret key of the scheme x. At times, we write $\mathsf{xct}(m)$, $\mathsf{xsk}(f)$ to make it explicit what is the encrypted message m and the associated function f; and write $\mathsf{xct}(k, m)$, $\mathsf{xsk}(k, f)$ to make explicit what is the key k they are generated from. We omit these details when they do not matter or are clear from the context.

2.1 Overview of Our FE Construction

Basic template of FE construction in prior works. We start with reviewing the basic template of FE construction in recent works [3, 11, 57]. FE allows one to generate so-called functional secret key $\mathsf{fesk}(f)$ associated with a function f that decrypts an encryption of a plaintext $\boldsymbol{x}$, $\mathsf{fect}(\boldsymbol{x})$ to $f(\boldsymbol{x})$. Security ensures that beyond the evaluation of the function f on $\boldsymbol{x}$, nothing is revealed about $\boldsymbol{x}$. For constructing $i\mathcal{O}$, it suffices to have an FE scheme whose security is guaranteed against adversaries seeing only *a single functional secret key*, for a function with long output $f : \{0,1\}^n \to \{0,1\}^m$ and where the ciphertexts are *sublinearly-compact* in the sense that its size depends sublinearly in the output length m.

Towards this, the basic idea is encrypting the message using a Homomorphic Encryption scheme HE, which produces the ciphertext $\mathsf{hect}(\boldsymbol{s}, \boldsymbol{x})$, where $\boldsymbol{s}$ is the secret key of HE. It is possible to publicly evaluate homomorphically any function f directly on the ciphertext to obtain an so-called output ciphertext $\mathsf{hect}(\boldsymbol{s}, f(\boldsymbol{x})) \leftarrow \mathsf{HEEval}(\mathsf{hect}, f)$, that encrypts the output $f(\boldsymbol{x})$. Then, we use another *much simpler* FE scheme to decrypt $\mathsf{hect}(\boldsymbol{s}, f(\boldsymbol{x}))$ so as to reveal $f(\boldsymbol{x})$ and nothing more. Using this paradigm, the computation of the function f is delegated to HE, while the FE only computes the decryption of HE. This is motivated by the fact that HE for arbitrary functions can be built from standard assumptions, while existing FE schemes is either not compact, in the sense that the ciphertext grows with the output size of the functions [75, 50], or are limited to basic functions — namely, degree-2 polynomials at most, [20, 44] for the public-key setting, [63, 14] for the private-key setting[6]Furthermore, known HE schemes have very simple decryption — for most of them, it is simply computing an inner product, then rounding. That is, decryption computes $\langle \mathsf{hect}_f, \boldsymbol{s} \rangle = p/2 \cdot f(\boldsymbol{x}) + \boldsymbol{e}_f \pmod p$ for some modulus p, where $\boldsymbol{s}$ is the secret key of HE, and $\boldsymbol{e}_f$ is a small, polynomially bounded error (for simplicity, in this overview, we assume w.l.o.g that $f(\boldsymbol{x}) \in \{0,1\}$). While there are FE schemes that support computing inner products [1, 4], sublinearly compact FE that also computes the rounding are currently our of reach. Omitting this rounding would reveal $f(\boldsymbol{x})$, but also $\boldsymbol{e}_f$,

[6] As mentioned in the introduction, partially hiding functional encryption allows to further strengthen the function class supported, by essentially adding computation on a public input, however computation on the private input is still limited to degree 2.

which hurts the security of HE. Instead, we will essentially realize an approximate version of the rounding — thereby hiding the noise $\boldsymbol{e}_f$.

A natural approach to hide the noises $\boldsymbol{e}_f$ is to use larger, smudging noises. Since $\boldsymbol{e}_f$ depends on the randomness used by HEEnc, and the function f, the smudging noises must be fresh for every ciphertext. Hard-wiring the smudging noise in the ciphertext, as done in [7], leads to non-succinct ciphertext, whose size grows linearly with the output size of the functions. Instead, we generate the smudging noises from a short seed, using a PRG. The latter must be simple enough to be captured by state of the art FE schemes.

Previous constructions use a weak pseudo-random generator, referred to as a noise generator NG, to generate many smudging noises $\boldsymbol{r} = \mathsf{NG}(\mathsf{sd})$ for hiding $\boldsymbol{e}_f$. To see how it works, suppose hypothetically that there is a noise generator computable by degree-2 polynomials. Then we can use $\mathsf{2FE}$, an FE scheme that support the generation of functional key for degree-2 polynomials, to compute $p/2 \cdot f(\boldsymbol{x}) + \boldsymbol{e}_f + \mathsf{NG}(\mathsf{sd})$, which reveals only $f(\boldsymbol{x})$ as desired. This gives a basic template of FE construction summarized below.

Basic Template of FE Construction (Intuition only, does not work)

$\mathsf{fesk}(f)$ contains : $\mathsf{2fsk}(g)$
$\mathsf{fect}(\boldsymbol{x})$ contains : $\mathsf{hect}(\boldsymbol{s}, \boldsymbol{x})$, $\mathsf{2fct}(\boldsymbol{s}||\mathsf{sd})$

The basic idea is using HE *with a one-time secret key* $\boldsymbol{s}$ *to perform the computation and using a simple FE for degree-2 polynomials,* $\mathsf{2FE}$*, to decrypt the output ciphertext and add a smudging noise generated via a noise generator* NG*. That is, we would like* $g(\boldsymbol{s}||\mathsf{sd}) = (p/2 \cdot f(\boldsymbol{x}) + \boldsymbol{e}_f + \mathsf{NG}(\mathsf{sd}))$*. However, there are many challenges to making this basic idea work.*

Unfortunately, to make the above basic idea work, we need to overcome a series of challenges. Below, we give an overview of the challenges, how we solve them using new tools, new techniques, and new assumptions, and how our solutions compare with previous solutions. In later subsections 2.2,2.3, and in the full version, we give more detail on our solutions.

Challenge 1: No Candidate Degree-2 Noise Generator. Several constraints are placed on the structure of the noise generators NG which renders their instantiation difficult.

- Minimal Degree. To use degree-2 FE to compute NG, the generator is restricted to have only degree 2 in the secret seed sd.
- Small (Poly-sized) Outputs. Existing degree-2 FE are implemented using pairing groups: They perform the degree-2 computation in the exponent of the groups, and obtain the output in the exponent of the target group. This means the output $p/2 \cdot f(\boldsymbol{x}) + \boldsymbol{e}_f + \mathsf{NG}(\mathsf{sd})$ resides in the exponent, and the only way to extract $f(\boldsymbol{x}) \in \{0, 1\}$ is via brute force discrete logarithm to

extract the whole $p/2 \cdot f(\boldsymbol{x}) + \boldsymbol{e}_f + \mathsf{NG}(\mathsf{sd})$. This in particular restricts NG to have polynomially bounded outputs.

Previous works [11, 57] used new assumptions that combine LWE with constant-degree polynomials over the integers (see discussion in the introduction) to instantiate the noise generator. The resulting generator do not have exactly degree 2, but "close" to degree 2 in following sense:

Degree "2.5" Noise Generator: $\mathsf{NG}(\mathsf{pubsd}, \mathsf{privsd})$ is a polynomial in a public seed pubsd and a private seed privsd both of length n', and has polynomial stretch. The seeds are jointly sampled $(\mathsf{pubsd}, \mathsf{privsd}) \leftarrow \mathcal{D}_{\mathsf{sd}}$ from some distribution and pubsd is made publc. Degree 2.5 means that NG has constant degree in pubsd and degree 2 in privsd.

Previous degree-2.5 noise generators produce small integer outputs, and can only satisfy certain weak pseudo-randomness property (as opposed to standard pseudorandomness). To get a flavor, consider the fact that the outputs of previous candidates are exactly the outputs of some constant-degree polynomials computed over the integers. Individual output elements are not uniformly distributed in any range, and two output elements that depend on the same seed element are noticably correlated. Hence, they are not pseudorandom or even pseudo-independent. In this work, our new assumption combines Learning With Binary Errors (LWBE) and constant-degree *Boolean* PRGs, and gives new degree-2.5 noise generators with *Boolean outputs* as follows:

- $\mathsf{pubsd} = \{\boldsymbol{c}_i = (\boldsymbol{a}_i, \boldsymbol{a}_i\boldsymbol{s}+e_i)\}_{i\in[n]}$: LWBE samples where $\boldsymbol{s}, \boldsymbol{a}_i \leftarrow \mathbb{Z}_p^{n^{0.5+\epsilon}}$, $e_i \leftarrow \{0,1\}$.
- $\mathsf{privsd} = \otimes(\boldsymbol{s}||-1)^{\lceil \frac{d}{2} \rceil}$: tensoring $(\boldsymbol{s}||-1)$ for $\lceil \frac{d}{2} \rceil$ times.
- $\mathsf{PRG}(\mathsf{pubsd}, \mathsf{privsd}) = G(\cdots||e_i = \langle \boldsymbol{c}_i, (\boldsymbol{s}||-1)\rangle||\cdots) = G(\boldsymbol{e})$, where G is a constant degree Boolean PRG.

When the PRG G has sufficient stretch $m \geq n^{\lceil \frac{d}{2} \rceil \cdot (0.5+\epsilon)+\rho}$ for some constant $\epsilon, \rho > 0$, our new generator has polynomial stretch $m = |\mathsf{pubsd}||\mathsf{privsd}|^{1+\epsilon'}$ for some ϵ' depending on ϵ, ρ. Constant-degree Boolean PRGs are qualitatively different from constant-degree polynomials over the integers and have been extensively studied. Furthermore, our new assumption implies that the outputs of our generator are *pseudo-random* – in other words, we obtain a *degree-2.5 Boolean* PRG.

Not surprisingly, the stronger security property of degree-2.5 PRG lets us significantly simplify the construction and security proof.

Challenge 2: How to Evaluate Degree 2.5 Polynomials? To evaluate our degree-2.5 Boolean PRG, we need an FE scheme that is more powerful than $\mathsf{2FE}$. The notion of Partially-Hiding Functional Encryption PHFE, originally introduced by [54] in the form of Partially Hiding Predicate Encryption (PHPE), fits exactly this task. As mentioned in introduction, PHFE strengthens the functionality of FE by allowing the ciphertext $\mathsf{phfct}(\boldsymbol{x}, \boldsymbol{y})$ to encode a public input $\boldsymbol{x}$, in addition to the usual private input $\boldsymbol{y}$. Decryption by a functional key $\mathsf{phfsk}(f)$ reveals $\boldsymbol{x}$

and $f(\boldsymbol{x}, \boldsymbol{y})$ and nothing else. The works of [11, 57] constructed *private-key* PHFE for computing degree-2.5 polynomials (i.e., constant degree in $\boldsymbol{x}$ and degree 2 in $\boldsymbol{y}$) from pairing groups. (Like $\mathsf{2FE}$, the output is still computed in the exponent of the target group.) This suffices for evaluating degree-2.5 noise generator or PRG in the FE construction outlined above. The only drawback is that since PHFE is private-key, the resulting FE is also private-key.

In this work, we give a new construction of PHFE from pairing groups that is 1) public-key and 2) supports arithmetic NC^1 computation on the public input — more specifically, $f(\boldsymbol{x}, \boldsymbol{y}) = \langle g(\boldsymbol{x}), q(\boldsymbol{y}) \rangle$ where g is computable by an arithmetic log-depth circuit and q is a degree-2 polynomial.

Theorem 2 (Public-key $(\mathsf{NC}^1, \textbf{deg-2})$-PHFE, Informal). *There is a construction of a public-key PHFE for arithmetic NC^1 public computation and degree-2 private computation from standard assumptions over asymmetric pairing groups.*

This new construction allows us to obtain public key FE directly. Furthermore, our construction supports the most expressive class of functions among all known FE schemes from standard assumptions; we believe this is of independent interests.

Challenge 3: How to Ensure Integrity? Now that we have replaced $\mathsf{2FE}$ with PHFE to compute degree-2.5 polynomials, the last question is how to ensure that PHFE decrypts only the right evaluated ciphertext hect_f (instead of any other ciphertext)? The function g we would like to compute via PHFE is $g(\boldsymbol{s}, \mathsf{pubsd}, \mathsf{privsd}) = \langle \mathsf{hect}_f, \boldsymbol{s} \rangle + \mathsf{NG}(\mathsf{pubsd}, \mathsf{privsd})$. The difficulty is that hect_f is unknown at key-generation time or at encryption time (as it depends on both f and $\mathsf{hect}(\boldsymbol{s}, \boldsymbol{x})$), and is too complex for PHFE to compute (as the homomorphic evaluation has high polynomial depth). To overcome this, we replace homormophic encryption with a *single-ciphertext* secret-key FE for polynomial size circuits with depth λ with *linear key generation*, denoted as ϵ-$\mathsf{1LGFE}$, which has the following special structure.

Single Ciphertext FE with Linear Key Generation

$\mathsf{PPGen}(1^\lambda)$	: generate public parameters pp
$\mathsf{Setup}(1^\lambda, \mathsf{pp})$	: generate master secret key $\boldsymbol{s} \in \mathbb{Z}_p^\lambda$
$\mathsf{Enc}(\mathsf{pp}, \boldsymbol{s})$	: generates a ciphertext ϵ-$\mathsf{1LGFE.ct}$
$\mathsf{KeyGen}(\mathsf{pp}, \boldsymbol{s}, f)$	: $\mathsf{pp}_f \leftarrow \mathsf{EvalPP}(\mathsf{pp}, f)$, $\boldsymbol{r} \leftarrow ([0, B-1] \cap \mathbb{Z})^m$, output f and secret key, ϵ-$\mathsf{1LGFE.sk}(f) = \langle \mathsf{pp}_f, \boldsymbol{s} \rangle - \boldsymbol{r}$
$\mathsf{Dec}(\epsilon$-$\mathsf{1LGFE.ct}, (f, \epsilon$-$\mathsf{1LGFE.sk}))$	: ϵ-$\mathsf{1LGFE.ct}_f \leftarrow \mathsf{EvalCT}(\epsilon$-$\mathsf{1LGFE.ct}, f)$ output $\frac{q}{2}\boldsymbol{y} + \boldsymbol{e}_f + \boldsymbol{r} \leftarrow \epsilon$-$\mathsf{1LGFE.ct} - \epsilon$-$\mathsf{1LGFE.sk}$, $\lvert\boldsymbol{e}_f\rvert_\infty \le B'$

The single-ciphertext FE has i) a key generation algorithm that is linear in the master secret key $\boldsymbol{s}$ and randomness $\boldsymbol{r}$, and ii) decryption first performs some computation on the ciphertext ϵ-$\mathsf{1LGFE.ct}$ to obtain an intermediate ciphertext ϵ-$\mathsf{1LGFE.ct}_f$, and then simply subtracts the secret key from ϵ-$\mathsf{1LGFE.ct}_f$, and obtains the output $\boldsymbol{y}$ perturbed by a polynomially-bounded noise.

We replace the ciphertext $\mathsf{hect}(\boldsymbol{s}, \boldsymbol{x})$ now with a ciphertext ϵ-$\mathsf{1LGFE.ct}(\boldsymbol{s}, \boldsymbol{x})$ of ϵ-$\mathsf{1LGFE}$. By the correctness and security of ϵ-$\mathsf{1LGFE}$, revealing ϵ-$\mathsf{1LGFE.sk}(f)$ only reveals the output $f(\boldsymbol{x})$. Hence, it suffices to use PHFE to compute the secret key. Thanks to the special structure of the key generation algorithm, this can be done in degree 2.5, using pseudoradnomness $\boldsymbol{r}$ expanded out via our degree-2.5 PRG. More concretely, PHFE computes the following degree-2.5 function g.

$$g(\boldsymbol{s}||\mathsf{pubsd}||\mathsf{privsd}) = \langle \mathsf{pp}_f, \boldsymbol{s} \rangle + \boldsymbol{r} = \epsilon\text{-}\mathsf{1LGFE.sk}(f), \qquad \texttt{// } g \text{ has degree 2.5}$$

$$\text{where } r_j = \sum_{k=0}^{\log B - 1} 2^k \mathsf{PRG}_{(j-1)\log B + k}(\mathsf{pubsd}, \mathsf{privsd}) \ .$$

One more technical caveat is that known pairing-based PHFE schemes actually compute the secret key ϵ-$\mathsf{1LGFE.sk}$ in the exponent of a target group element, which we denote by $[\epsilon\text{-}\mathsf{1LGFE.sk}]_T$, where for any exponent $a \in \mathbb{Z}_p$, $[a]_T = g_T^a$ for a generator g_T. Thanks to the special structure of the decryption algorithm of ϵ-$\mathsf{1LGFE}$ — namely, it is linear in ϵ-$\mathsf{1LGFE.sk}$ — these group elements are sufficient for decryption. A decryptor can first compute ϵ-$\mathsf{1LGFE.ct}_f$ from ϵ-$\mathsf{1LGFE.ct}(\boldsymbol{s}, \boldsymbol{x})$ and f in the clear, then perform the decryption by subtracting $[\epsilon\text{-}\mathsf{1LGFE.ct}_f - \epsilon\text{-}\mathsf{1LGFE.sk}]_T$ in the exponent. This gives $[p/2 \cdot f(\boldsymbol{x}) + \boldsymbol{e}_f + \boldsymbol{r}]_T$, whose exponent $p/2 \cdot f(\boldsymbol{x}) + \boldsymbol{e}_f + \boldsymbol{r}$ can be extracted by enumrating all possible $\boldsymbol{e}_f + \boldsymbol{r}$, which are of polynomial size, and $f(\boldsymbol{x}) \in \{0, 1\}$.

Our single-ciphertext FE with linear key generation is essentially the same notion as that of Special Homomorphic Encryption (SHE) used in [3, 64]. SHE are homomorphic encryption with a special decryption equation $\mathsf{hect}_f - \langle \mathsf{pp}_f, \boldsymbol{s} \rangle = p/2 \cdot f(\boldsymbol{x}) + \boldsymbol{e}_f$ where pp_f (as in ϵ-$\mathsf{1LGFE}$) can be computed efficiently from public parameters pp and f. We think it is more accurate to view this object as a

functional encryption scheme, since what the special decryption equation gives is exactly a functional key $\langle \mathsf{pp}_f, \boldsymbol{s} \rangle + \boldsymbol{r}$ where $\boldsymbol{r}$ are smudging noises for hiding $\boldsymbol{e}_f$ to guarantee that only $p/2 \cdot f(\boldsymbol{x})$ is revealed.

Viewing this through the lens of FE brought us a significant benefit. Previous works constructed SHE by modifying the Brakerski-Vankuntanathan FHE scheme [33], but are limited to supporting NC^1 computations based on RLWE [7], and NC_0 based on LWE [7, 64]. Instead, the FE lens led us to search for ideas in the predicate encryption literature. We show how to construct ϵ-1LGFE for polynomial sized circuits with depth bounded by λ from LWE by modifying the predicate encryption scheme of [54]. This new construction allowed us to construct FE for polynomial sized circuits with depth bounded by λ directly without invoking any bootstrapping theorem from weaker function classes.

Theorem 3 (ϵ-1LGFE from LWE, informal). *There is a construction of a single-ciphertext FE for polynomial size circuits of depth λ with linear key generation as described above, from LWE.*

In summary, putting all the pieces together, our construction of FE for polynomial size circuits with depth λ is depicted below. Comparing with previous constructions, it enjoys several features: 1) it is public key, 2) it can be based on the polynomial-hardness of underlying assumptions, 3) it has simpler proofs (e.g., no bootstrapping theorem).

Our FE Construction

$\mathsf{fesk}(f)$ contains : $\mathsf{phfsk}(g)$
$\mathsf{fect}(\boldsymbol{x})$ contains : $\epsilon\text{-}\mathsf{1LGFE.ct}(\boldsymbol{s}, \boldsymbol{x})$ $\mathsf{phfct}(\boldsymbol{s}||\mathsf{pubsd}||\mathsf{privsd})$

$\mathsf{FEDec}(\mathsf{fect}, (f, \mathsf{fesk}))$: $[\epsilon\text{-}\mathsf{1LGFE.sk}]_T \leftarrow \mathsf{PHFEDec}(\mathsf{phfct}, \mathsf{phfsk})$
$\epsilon\text{-}\mathsf{1LGFE.ct}_f \leftarrow \mathsf{EvalCT}(\epsilon\text{-}\mathsf{1LGFE.ct}, f)$
$[\boldsymbol{y} + \boldsymbol{e}_f + \boldsymbol{r}]_T = \epsilon\text{-}\mathsf{1LGFE.ct}_f - [\epsilon\text{-}\mathsf{1LGFE.sk}]_T$
extract $\boldsymbol{y} + \boldsymbol{e}_f + \boldsymbol{r}$ and round to recover $\boldsymbol{y}$

The basic idea is using PHFE *to compute a* ϵ-1LGFE *secret key* $\epsilon\text{-}\mathsf{1LGFE.sk}(f)$ *in the exponent of the target group, and then decrypting the ciphertext* $\epsilon\text{-}\mathsf{1LGFE.ct}(\boldsymbol{s}, \boldsymbol{x})$ *to reveal* $f(\boldsymbol{x})$ *only.*

The only aspect of our construction that we have not discussed explicitly is how to deal with the fact that the pseudorandom smudging error is of polynomial size, and therefore reveals a $1/poly$ amount of information. We thus need to amplify security, but because the source of our error is so simple, we are able to achieve this amplification in a simple and direct construction (found in the full version) that avoids any need to use hard-core measures or any other such sophisticated and/or delicate amplification technology.

2.2 Instantiating Our Assumption

To instantiate our assumption, we need to *choose a degree d PRG with a stretch more than* $n^{\lceil \frac{d}{2} \rceil \cdot (0.5+\delta)+\rho}$. The good news is that there is a rich body of literature

on both ingredients of our assumption that existed way before our work to guide the choice. Binary LWE was first considered by [18] and then by [68, 8, 35, 36]. Goldreich PRGs have been studied even before that. There are many prior works spanning areas in computer science devoted to cryptanalysis of these objects from lattice reduction algorithms and symmetric-key cryptanalysis, to algebraic algorithm tools such as the Gröbner basis algorithm and attacks arising from the Constraint Satisfaction Problem and Semi-Definite Programming literature. Guided by them, we list three candidates below. In the full-version [45], we survey many of these attack algorithms, and we compute approximate running times of the attacks arising out of these algorithms on our candidates. For the parameters we choose, all those attacks are subexponential time.

A Goldreich's PRG G is defined by a predicate $P : \{0,1\}^{\ell'} \to \{0,1\}$, where ℓ' is the locality of the PRG, and a bipartiate input-output dependency graph Λ, which specifies for every output index $j \in [m]$, the subset $\Lambda(j) \subset [n]$ of input indexes of size ℓ' it depends on – the j'th output bit is simply set to $G(j) = P(\Lambda(j))$. Hence the degree of the PRG G is identical to the degree of the predicate P. Usually, the input-output dependency graph Λ is chosen at random, and the non-trivial part lies in choosing the predicate P.

Instantiation 1. The first instantiation is that of the predicate XORMAJ, which is a poplular PRG predicate [17, 40].

$$\mathsf{XORMAJ}_{\ell,\ell}(x_1 \ldots, x_{2\ell}) = \oplus_{i\in[\ell]} x_i \oplus \mathsf{MAJ}(x_{\ell+1}, \ldots, x_{2\ell}).$$

The predicate above has a degree of $2\cdot\ell$; thus, our construction require expansion $m > n^{\frac{\ell}{2}+\ell\delta+\rho}$. The predicate is $\ell+1$ wise independent and thus it provably resists subexponential time SoS refutation attacks when $m(n) \leq n^{\frac{\ell+1}{2}-c}$ for $c > 0$ [61]. All other known attacks that we consider and even the algebraic attacks when instantiated in our combined assumption require subexponential time. We refer the reader to the full-version [45] for a detailed discussion.

Instantiation 2. An slightly unsatisfactory aspect of the XORMAJ predicate is that the lower bound on the stretch of the PRG instantiated by XORMAJ for it to be useful in our FE construction is $> n^{\frac{\ell}{2}+\delta'}$, whereas the upper bound on the stretch to withstand existing attacks is very close $\leq n^{\frac{\ell+1}{2}-c}$, leaving only a tiny margin to work with. This motivates us to we consdier predicates with degree lower than the locality. One such predicate was analyzed in [67] for stretch upto $n^{1.25-c}$ for $c > 0$:

$$\mathsf{TSPA}(x_1, x_2, x_3, x_4, x_5) = x_1 \oplus x_2 \oplus x_3 \oplus ((x_2 \oplus x_4) \wedge (x_3 \oplus x_5)).$$

What is nice about this predicate is that, it has locality 5 but only degree 3; thus, our construction only require expansion $m > n^{\lceil\frac{3}{2}\rceil(0.5+\epsilon)+\rho} = n^{1+2\epsilon+\rho}$. In [67], it was proven that the PRG istantiated with TSPA resists subexponential time $\mathbb{F}_2$ linear and SoS attacks. We present analysis against other attacks in the full-version [45], all taking subexponential time.

Instantiation 3. We present a degree reduction transformation that takes as input a non-linear predicate $g : \{0,1\}^k \to \{0,1\}$ and constructs a predicate P.

$$\mathsf{P}_g(x_1 \ldots, x_{2k+1}) = \oplus_{i \in [k+1]} x_i \oplus g(x_{k+2} \oplus x_2, \ldots, x_{2k+1} \oplus x_{k+1}).$$

We show in the full version [45] that the predicate above has a locality of $2k+1$ but a degree equal to $k+1$; thus, our construction requires expansion $m > n^{\lceil \frac{k+1}{2} \rceil (0.5+\epsilon)+\rho}$. The predicate is also $k+1$ wise independent. We show that all known attacks run in subexponential time even when the stretch is bounded by $m \leq n^{\frac{k+1}{2}-\delta}$ for some $\delta > 0$. Thanks to the gap between the locality and degree, we now have a very large margin between the lower and upper bounds on the stretch. Hence, our work motivates the interesting question of studying such predicates.

2.3 Single Ciphertext Functional Encryption with Linear Key Generation

We describe our construction of a single-ciphertext (secret-key) FE scheme for all polynomial-sized circuits with depth bounded by λ, that have the simple structure outlined in Section 2, denoted as ϵ-$\mathsf{1LGFE}$, from LWE. In particular, the key generation and decryption algorithms have the following form, where $\boldsymbol{s}$ is the master secret key and pp is the public parameters.

$$\begin{aligned}
&\mathsf{KeyGen}(\mathsf{pp}, \boldsymbol{s}, f) && : \mathsf{pp}_f \leftarrow \mathsf{EvalPP}(\mathsf{pp}, f)\ ,\ \boldsymbol{r} \leftarrow ([0, B-1] \cap \mathbb{Z})^m, \\
& && \text{output } f \text{ and secret key } \epsilon\text{-}\mathsf{1LGFE.sk}(f) = \langle \mathsf{pp}_f, \boldsymbol{s} \rangle - \boldsymbol{r} \\
&\mathsf{Dec}(\epsilon\text{-}\mathsf{1LGFE.ct}, (f, \epsilon\text{-}\mathsf{1LGFE.sk})) && : \epsilon\text{-}\mathsf{1LGFE.ct}_f \leftarrow \mathsf{EvalCT}(\epsilon\text{-}\mathsf{1LGFE.ct}, f) \\
& && \text{output } \tfrac{q}{2}\boldsymbol{y} + \boldsymbol{e}_f + \boldsymbol{r} \leftarrow \epsilon\text{-}\mathsf{1LGFE.ct} - \epsilon\text{-}\mathsf{1LGFE.sk},\ |\boldsymbol{e}_f|_\infty \leq B'
\end{aligned}$$

Importantly, decryption recovers a perturbed output where the error $\boldsymbol{e}_f + \boldsymbol{r}$ is polynomially bounded. As mentioned before, this object is essentially the same as the notion of Special Homomorphic Encryption (SHE) in the literature [7, 64]. Previous SHE schemes are constructed by modifying existing homomorphic encryption schemes of [33, 31]. These constructions are recursive and quite complex, and the overhead due to recursion prevents them from supporting computations beyond NC^1. In this work, viewing through the FE lens, we search the literature of predicate encryption, and show how to modify the predicate encryption scheme of [54] (GVW) to obtain single-ciphertext FE with the desired structure. The GVW predicate encryption provide us with a single-ciphertext encryption scheme with the following properties:

- The public parameter generation algorithm PPGen samples a collection of random LWE matrices $\boldsymbol{A}_i, \boldsymbol{B}_j \leftarrow \mathbb{Z}_p^{n \times m}$, and sets the public parameters to $\mathsf{pp} = (\{\boldsymbol{A}_i\}, \{\boldsymbol{B}_j\})$.
- The setup algorithm Setup samples a master secret key constaining an LWE secret $\boldsymbol{s} \leftarrow \chi^n$ drawn from the noise distribution χ.

– The encryption algorithm to encrypt $\boldsymbol{x}$, generates a ciphertext $\mathsf{hect}(\boldsymbol{x})$ containing two sets of LWE samples of form $\boldsymbol{c}_i = \boldsymbol{s}^T\boldsymbol{A}_i + \widehat{\boldsymbol{x}}_i\boldsymbol{G} + \boldsymbol{e}_i$ and $\boldsymbol{d}_j = \boldsymbol{s}^T\boldsymbol{B}_j + \widehat{k}_j\boldsymbol{G} + \boldsymbol{e}'_j$, where $\boldsymbol{G} \in \mathbb{Z}_p^{n\times m}$ is the gadget matrix, vk is a freshly sampled secret key of a homomorphic encryption scheme, and $\boldsymbol{e}_i, \boldsymbol{e}'_j \leftarrow \chi^m$ are LWE noises. Furthermore, $\widehat{\boldsymbol{x}}_i$ is the i'th bit of a homomorphic encryption ciphertext of $\boldsymbol{x}$ under key $\boldsymbol{k}$.
– The predicate encryption scheme of [54] provides two homomorphic procedures: The EvalCT procedure homomorphically evaluate f on $\{\boldsymbol{c}_i, \boldsymbol{A}_i\}$ and $\{\boldsymbol{d}_j, \boldsymbol{B}_j\}$ to obtain $\boldsymbol{c}_f$, and the EvalPP seperately homormorphically evaluates on $\{\boldsymbol{A}_i\}$ and $\{\boldsymbol{B}_i\}$ to obtain $\boldsymbol{A}_f$.
– The homomorphic evaluation outcomes $\boldsymbol{c}_f, \boldsymbol{A}_f$, has the property that the first coordinate $\boldsymbol{c}_{f,1}$ of $\boldsymbol{c}_f$ and the first column $\boldsymbol{A}_{f,1}$ of $\boldsymbol{A}_f$ satisfy the special decryption equation.

$$\boldsymbol{c}_{f,1} - \boldsymbol{s}^T\boldsymbol{A}_{f,1} = f(\boldsymbol{x})\lfloor p/2\rceil + e_f \mod p$$

The above described encryption scheme almost gives the FE scheme we want except for the issue that it has super-polynomially large decryption error e_f. Thus, we turn to reducing the norm of the decryption error, by applying the rounding (or modulus switch) technique in the HE literature [31]. Namely, to reduce the error norm by a factor of p/q for a $q < p$, we multiply $\boldsymbol{c}_{f,1}$ and $\boldsymbol{A}_{f,1}$ with q/p over the reals and then round to the nearest integer component wise. The rounding results satisfy the following equation

$$\lfloor\frac{q}{p}\boldsymbol{c}_{f,1}\rceil - \boldsymbol{s}^T\lfloor\frac{q}{p}\boldsymbol{A}_{f,1}\rceil = f(\boldsymbol{x})\lfloor q/2\rceil + \lfloor\frac{q}{p}e_f\rceil + \mathsf{error} \mod p$$

where the rounding error error is bounded by $|\mathsf{hesk}|_1 + O(1)$, which is polynomially bounded as the secret is sampled from the LWE noise distribution instead of uniformly.

We are now ready to instantiate the FE scheme we want. It uses the same public parameter generation, setup, and encryption algorithm. Now to generate a functional key for f, it first computes $A_f \leftarrow \mathsf{EvalPP}(\{\boldsymbol{A}_i\}, \{\boldsymbol{B}_j\})$ and sets $\mathsf{pp}_f = \lfloor\frac{q}{p}\boldsymbol{A}_{f,1}\rceil$, and then outputs a functional key $\epsilon\text{-}\mathsf{1LGFE.sk} = \langle \mathsf{pp}_f \boldsymbol{s}\rangle - \boldsymbol{r}$ where $\boldsymbol{r}$ is a random vector of smudging noises of sufficiently large but still polynomially bounded magnitude. The decryption algorithm decrypts a ciphertext $\epsilon\text{-}\mathsf{1LGFE.ct} = (\{\boldsymbol{c}_i\}, \{\boldsymbol{d}_j\})$ using a functional key $\epsilon\text{-}\mathsf{1LGFE.sk}$ as follows: It first computes $\boldsymbol{c}_f \leftarrow \mathsf{EvalPP}(\{\boldsymbol{A}_i, \boldsymbol{c}_i\}, \{\boldsymbol{B}_j, \boldsymbol{d}_j\})$, and sets $\epsilon\text{-}\mathsf{1LGFE.ct}_f = \lfloor\frac{q}{p}\boldsymbol{c}_{f,1}\rceil$, it then subtracts $\epsilon\text{-}\mathsf{1LGFE.sk}$ from it, yielding $f(\boldsymbol{x})\lfloor q/2\rceil + \lfloor\frac{q}{p}e_f\rceil + \mathsf{error} + \boldsymbol{r}$ as desired.

3 Preliminaries

In this section, we describe preliminaries that are useful for rest of the paper. We denote the security parameter by λ. For any distribution $\mathcal{X}$, we denote by $x \leftarrow \mathcal{X}$ (or $x \leftarrow_{\mathrm{R}} \mathcal{X}$) the process of sampling a value x from the distribution $\mathcal{X}$.

Similarly, for a set X we denote by $x \leftarrow X$ (or $x \leftarrow_{\mathrm{R}} X$) the process of sampling x from the uniform distribution over X. For an integer $n \in \mathbb{N}$ we denote by $[n]$ the set $\{1, .., n\}$. A function $\mathsf{negl} : \mathbb{N} \to \mathbb{R}$ is negligible if for every constant $c > 0$ there exists an integer N_c such that $\mathsf{negl}(\lambda) < \lambda^{-c}$ for all $\lambda > N_c$.

By $\approx_c$ we denote the standard polynomial time computational indistinguishability. We say that two ensembles $\mathcal{X} = \{\mathcal{X}_\lambda\}_{\lambda\in\mathbb{N}}$ and $\mathcal{Y} = \{\mathcal{Y}_\lambda\}_{\lambda\in\mathbb{N}}$ are $(s(\lambda), \epsilon(\lambda))-$ indistinguishable if for every adversary $\mathcal{A}$ (modeled as a circuit) of size bounded by $s(\lambda)$ it holds that: $\left| \Pr_{x\leftarrow\mathcal{X}_\lambda}[\mathcal{A}(1^\lambda, x) = 1] - \Pr_{y\leftarrow\mathcal{Y}_\lambda}[\mathcal{A}(1^\lambda, y) = 1] \right| \leq \epsilon(\lambda)$ for every sufficiently large $\lambda \in \mathbb{N}$.

For a field element $a \in \mathbb{F}_{\mathsf{prmtr}}$ represented in $[-p/2, p/2]$, we say that $a \in [-B, B]$ for some positive integer B if its representative in $[-p/2, p/2]$ lies in $[-B, B]$.

Throughout, when we refer to polynomials in security parameter, we mean constant degree polynomials that take positive value on non negative inputs. We denote by $\mathsf{poly}(\lambda)$ an arbitrary polynomial in security parameter satisfying the above requirements of non-negativity.

3.1 Pairing Groups

Throughout the paper, we use a sequence of asymmetric prime-order pairing groups:

$$\mathcal{G} = \{(p_\lambda,\, \mathbf{G}_{\lambda,1}, \mathbf{G}_{\lambda,2}, \mathbf{G}_{\lambda,T},\, P_{\lambda,1}, P_{\lambda,2}, P_{\lambda,T},\, e_\lambda)\}_{\lambda\in\mathbb{N}},$$

where for all $s \in \{1, 2, T\}$, $(\mathbf{G}_{\lambda,s}, +)$ is an cyclic group (for which we use additive notation) of order $p_\lambda = 2^{\lambda^{\Theta(1)}}$. $\mathbf{G}_{\lambda,1}$ and $\mathbf{G}_{\lambda,2}$ are generated by $P_{\lambda,1}$ and $P_{\lambda,2}$ respectively, and $e : \mathbf{G}_{\lambda,1} \times \mathbf{G}_{\lambda,2} \to \mathbf{G}_T$ is a non-degenerate bilinear map, that is, satisfying $e_\lambda(aP_{\lambda,1}, bP_{\lambda,2}) = abP_T$ for all integers $a, b \in \mathbb{Z}_p$, where $P_T = e(P_{\lambda,1}, P_{\lambda,2})$ is a generator of $\mathbf{G}_{\lambda,T}$. We require the group operations as well as the pairing operation to be efficiently computable. The rest of the paper will refer to this sequence of bilinear pairing groups, and the corresponding sequence of prime orders of the groups $\{p_\lambda\}_{\lambda\in\mathbb{N}}$. In the full version [45], we describe the assumptions bilateral DLIN and SXDH over such groups, which we use for our construction.

4 Functional Encryption Definitions

We denote by $\mathcal{F} = \cup_{n,d,\ell,\mathsf{size}\in\mathsf{poly}} \left(\{\mathcal{F}_{\lambda,n(\lambda),d(\lambda),\ell(\lambda),\mathsf{size}(\lambda)}\}_{\lambda\in\mathbb{N}}\right)$ an abstract function class, which is parameterised by $\lambda \in \mathbb{N}$ and four polynomials $n(\lambda), d(\lambda), \ell(\lambda), \mathsf{size}(\lambda))$. We call prmtr the tuple $(n, d, \ell, \mathsf{size})$. In this abstract class, every function $f \in \mathcal{F}_{\lambda,\mathsf{prmtr}}$ takes an input from $\mathcal{X}_{\lambda,\mathsf{prmtr}} \times \mathcal{Y}_{\lambda,\mathsf{prmtr}}$ and outputs in $\mathcal{Z}_{\lambda,\mathsf{prmtr}}$. We will specify what the exact denotes in the exact constructions. Two specific instantiations of those classes are described below:

– The function class $\mathcal{F}^{\mathsf{CIRC}}_{\lambda,\mathsf{prmtr}}$: Here $\mathcal{Y}_{\lambda,\mathsf{prmtr}}$ consists of $\{0,1\}^n$, $\mathcal{X}_{\lambda,\mathsf{prmtr}}$ is empty, $\mathcal{Z}_{\lambda,\mathsf{prmtr}} = \{0,1\}^\ell$. This family consists of Boolean circuits of depth d and size size.
– The function class $\mathcal{F}^{\mathsf{PHFE}}_{\lambda,\mathsf{prmtr}}$: Here $\mathcal{X}_{\lambda,\mathsf{prmtr}} = \mathcal{Y}_{\lambda,\mathsf{prmtr}} = \mathbb{Z}_{p_\lambda}^{O(n)}$ where p_λ is the prime order for the group $\mathcal{G}_\lambda$. The class consists of certain kinds of arithmetic cicuits over $\mathbb{Z}_p$. We describe the exact class later when we need it.

Here we provide the relevant definition regarding functional encryption (FE) and partially-hiding FE (PHFE) along with several notions of efficiency and security properties. FE corresponds to the particular case where the public part of the message (referred to as $\mathcal{X}_{\lambda,\mathsf{prmtr}}$ below) is empty.

Definition 1. *(Syntax of a PHFE Scheme.) A partially-hiding functional encryption scheme,* PHFE*, for a functionality* $\{\mathcal{F}_{\lambda,\mathsf{prmtr}} : \mathcal{X}_{\lambda,\mathsf{prmtr}} \times \mathcal{Y}_{\lambda,\mathsf{prmtr}} \to \mathcal{Z}_{\lambda,\mathsf{prmtr}}\}_{\lambda,\mathsf{prmtr}}$*, consists of the following PPT algorithms:*

– $\mathsf{PPGen}(1^\lambda, \mathsf{prmtr})$: *Given as input the security parameter* 1^λ *and additional parameters* $\mathsf{prmtr} = (n, d, \ell, \mathsf{size})$*, it outputs a string* pp*. We assume that* pp *is implicitly given as input to all the algorithms below.*
– $\mathsf{Setup}(\mathsf{pp})$*: Given as input* pp*, it outputs a public key* pk *and a master secret key* msk*.*
– $\mathsf{Enc}(\mathsf{pk}, (x, y))$*: Given as input the public key* pk *and a message* (x, y) *with public part* $x \in \mathcal{X}_{\lambda,\mathsf{prmtr}}$ *and private part* $y \in \mathcal{Y}_{\lambda,\mathsf{prmtr}}$*, outputs the ciphertext* ct *along with the input* x*.*
– $\mathsf{KeyGen}(\mathsf{msk}, f)$*: Given as input the master secret key* msk *and a function* $f \in \mathcal{F}_{\lambda,\mathsf{prmtr}}$*, it outputs a functional decryption key* sk_f*.*
– $\mathsf{Dec}(\mathsf{sk}_f, (x, \mathsf{ct}))$*: Given a functional decryption key* sk_f *and a ciphertext* (x, ct)*, it deterministically outputs a value* z *in* $\mathcal{Z}_{\lambda,\mathsf{prmtr}}$*, or* $\perp$ *if it fails.*

Remark 1. (On Secret Key Schemes.) An FE scheme is said to be secret-key is pk is empty, and the encryption algorithm takes as additional input the master secret key msk.

Remark 2. (On FE vs PHFE.) The syntax of FE is identical to PHFE described above except that for all $\lambda \in \mathbb{N}$, the set $\mathcal{X}_{\lambda,\mathsf{prmtr}} = \emptyset$, that is, all the input remains private.

Definition 2. *(Correctness.) A Partially hiding FE scheme* PHFE *for the functionality* $\mathcal{F} = \{\mathcal{F}_{\lambda,\mathsf{prmtr}}\}_{\lambda,\mathsf{prmtr}}$ *is correct if for security parameter* $\lambda \in \mathbb{N}$ *and every polynomials* $n, d, \ell, \mathsf{size}$ *there exists a negligible function* $\mathsf{negl}(\lambda)$ *such that for all messages* $(x, y) \in \mathcal{X}_{\lambda,\mathsf{prmtr}} \times \mathcal{Y}_{\lambda,\mathsf{prmtr}}$ *and all functions* $f \in \mathcal{F}$*, we have:*

$$\Pr\left[\begin{array}{c} \mathsf{pp} \leftarrow \mathsf{PPGen}(1^\lambda, \mathsf{prmtr}) \\ (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(\mathsf{pp}) \\ (x, \mathsf{ct}) \leftarrow \mathsf{Enc}(\mathsf{pk}, (x, y)) \\ \mathsf{sk}_f \leftarrow \mathsf{KeyGen}(\mathsf{sk}, f) \\ \mathsf{Dec}(\mathsf{sk}_f, x, \mathsf{ct})) \neq f(x, y) \end{array}\right] \leq \mathsf{negl}(\lambda).$$

Now we give the security notions for PHFE and FE.

4.1 Security Definition

We discuss two security notions. First, for any constant $\epsilon \in (0,1]$, we present the notion of ϵ-simulation security below:

Definition 3 (ϵ-simulation security). *For all $\epsilon \in (0,1]$, we say a PHFE scheme for the functionality $\mathcal{F} = \{\mathcal{F}_{\lambda,\mathsf{prmtr}}\}_{\lambda,\mathsf{prmtr}}$ denoted by PHFE is ϵ-simulation secure if there exists a (possibly stateful) PPT simulator $\mathcal{S} = (\widetilde{\mathsf{Setup}}, \widetilde{\mathsf{Enc}}, \widetilde{\mathsf{KeyGen}})$ such that for all stateful PPT adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, there exists a negligible function negl such that for all security parameters $\lambda \in \mathbb{N}$, all polynomials $\mathsf{prmtr} = (n, d, \ell, \mathsf{size})$, we have:*

$$\mathsf{adv}^{\mathsf{SIM}}_{\mathsf{PHFE},\mathcal{A}}(1^\lambda, \mathsf{prmtr}) := |\Pr[1 \leftarrow \mathsf{Real}^{\mathsf{PHFE}}_{\mathcal{A}}(1^\lambda, \mathsf{prmtr})] - \Pr[1 \leftarrow \mathsf{Ideal}^{\mathsf{PHFE}}_{\mathcal{A},\mathcal{S}}(1^\lambda, \mathsf{prmtr})]| < \mathsf{negl}(\lambda),$$

where the experiments $\mathsf{Real}^{\mathsf{PHFE}}_{\mathcal{A}}(1^\lambda)$ and $\mathsf{Ideal}^{\mathsf{PHFE}}_{\mathcal{A},\mathcal{S}}(1^\lambda)$ are defined below. The differences between these two experiments are highlighted in red.

$\underline{\mathsf{Real}^{\mathsf{PHFE}}_{\mathcal{A}}(1^\lambda, \mathsf{prmtr})}$*:*
$(x^*, y^*) \in \mathcal{X}_{\lambda,\mathsf{prmtr}} \times \mathcal{Y}_{\lambda,\mathsf{prmtr}}, (f_j \in \mathcal{F}_{\lambda,\mathsf{prmtr}})_{j\in[Q_{\mathsf{sk}}]} \leftarrow \mathcal{A}_1(1^\lambda)$
$\mathsf{pp} \leftarrow \mathsf{PPGen}(1^\lambda, \mathsf{prmtr})$
$(\mathsf{pk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(\mathsf{pp})$
$(x^*, \mathsf{ct}^*) \leftarrow \mathsf{Enc}(\mathsf{pk}, (x^*, y^*))$
$\forall j \in [Q_{\mathsf{sk}}]$: $\mathsf{sk}_{f_j} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, f_j)$
$\alpha \leftarrow \mathcal{A}_2(\mathsf{pp}, \mathsf{pk}, (\mathsf{sk}_{f_j})_{j\in Q_{\mathsf{sk}}}, x^*, \mathsf{ct}^*)$
Output α.

$\underline{\mathsf{Ideal}^{\mathsf{PHFE}}_{\mathcal{A},\mathcal{S}}(1^\lambda, \mathsf{prmtr})}$*:*
$(x^*, y^*) \in \mathcal{X}_{\lambda,\mathsf{prmtr}} \times \mathcal{Y}_{\lambda,\mathsf{prmtr}}, (f_j \in \mathcal{F}_{\lambda,\mathsf{prmtr}})_{j\in[Q_{\mathsf{sk}}]} \leftarrow \mathcal{A}_1(1^\lambda)$
$\mathsf{pp} \leftarrow \mathsf{PPGen}(1^\lambda, \mathsf{prmtr})$
$(\widetilde{\mathsf{pk}}, \mathsf{td}) \leftarrow \widetilde{\mathsf{Setup}}(\mathsf{pp})$, $\omega \leftarrow \mathsf{Sample}(x^*, y^*, (f_j)_{j\in[Q_{\mathsf{sk}}]})$
$(x^*, \widetilde{\mathsf{ct}^*}) \leftarrow \widetilde{\mathsf{Enc}}(\mathsf{td}, \omega)$
$\forall j \in [Q_{\mathsf{sk}}] : \widetilde{\mathsf{sk}_{f_j}} \leftarrow \widetilde{\mathsf{KeyGen}}(\mathsf{td}, f_j, \omega)$
$\alpha \leftarrow \mathcal{A}_2\left(\mathsf{pp}, \widetilde{\mathsf{pk}}, (\widetilde{\mathsf{sk}}_{f_j})_{j\in Q_{\mathsf{sk}}}, x^*, \widetilde{\mathsf{ct}^*}\right)$
Output α.

The algorithm Sample*, given as input the tuple $\left(x^*, (f_j, f_j(x^*, y^*))_{j\in[Q_{\mathsf{sk}}]}\right)$, flips a biased coin. If the outcome is tails (which happens with probability ϵ over the coin flip), then it outputs $\omega = \left(x^*, (f_j, f_j(x^*, y^*))_{j\in[Q_{\mathsf{sk}}]}\right)$. If the outcome is heads (which happens with probability $1-\epsilon$ over the coin flip), then it outputs $\omega = \left(x^*, y^*(f_j)_{j\in[Q_{\mathsf{sk}}]}\right)$.*

Remark 3 (Standard simulation security). If $\epsilon = 1$, the algorithm Sample always outputs $\omega = (x^*, (f_j, f_j(x^*, y^*))_{j\in[Q_{\mathsf{sk}}]})$, which corresponds to the standard simulation security definition.

Remark 4 (Secret-Key schemes). This definition can be easily adapted to a secret-key scheme simply by having the encryption algorithm get the additional input msk.

Remark 5 (Subexponential security). If $\epsilon = 1$, and the negl above is $2^{-\lambda^{\Omega(1)}}$, then the scheme is said to satisfy subexponential security.

Remark 6 (Number of functional decryption keys). We say a a scheme is many-key secure if security holds for any polynomial Q_{sk}, and one-key secure if $Q_{\mathsf{sk}} = 1$. When we do not specify it explicitly, we mean one-key security.

We also give an indistinguishability-based security definition.

Definition 4 (IND security). *We say an FE scheme* FE *for functionality* $\mathcal{F} = \{\mathcal{F}_{\lambda,\mathsf{prmtr}}\}_{\lambda\in\mathbb{N}}$ *is IND secure if for all stateful PPT adversaries* $\mathcal{A}$*, all polynomial parameters* $\mathsf{prmtr} = (n, d, \ell, \mathsf{size})$ *there exists a negligible function* negl *such that , we have:*

$$\mathsf{adv}^{\mathsf{IND}}_{\mathsf{FE},\mathcal{A}}(\lambda) := 2 \cdot |1/2 - \Pr[1 \leftarrow \mathsf{IND}^{\mathsf{FE}}_{\mathcal{A}}(1^\lambda, \mathsf{prmtr})]| < \mathsf{negl}(\lambda),$$

where the experiment $\mathsf{IND}^{\mathsf{FE}}_{\mathcal{A}}(1^\lambda, \mathsf{prmtr})$ *is defined below.*

$\underline{\mathsf{IND}^{\mathsf{FE}}_{\mathcal{A}}(1^\lambda, \mathsf{prmtr})}$*:*
$\{x^i_0, x^i_1\}_{i\in[Q_{\mathsf{ct}}]}, \{f^j\}_{j\in[Q_{\mathsf{sk}}]} \leftarrow \mathcal{A}(1^\lambda)$
$\mathsf{pp} \leftarrow \mathsf{PPGen}(1^\lambda, \mathsf{prmtr})$
Where $\forall i \in [Q]$*:* $x^i_0, x^i_1 \in \mathcal{Y}_{\lambda,\mathsf{prmtr}}$*,* $\forall j \in [Q_{\mathsf{sk}}]$*:* $f^j \in \mathcal{F}_{\lambda,\mathsf{prmtr}}$
$(\mathsf{pk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(\mathsf{pp})$*,* $b \leftarrow_R \{0, 1\}$
$\forall i \in [Q_{\mathsf{ct}}] : \mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{pk}, x^i_b)$*,* $\forall j \in [Q_{\mathsf{sk}}] : \mathsf{sk}_j \leftarrow \mathsf{KeyGen}(\mathsf{msk}, f^j)$
$b' \leftarrow \mathcal{A}(\{\mathsf{ct}_i\}_{i\in[Q_{\mathsf{ct}}]}, \{\mathsf{sk}_j\}_{j\in[Q_{\mathsf{sk}}]}, \mathsf{pk})$
Return 1 *if* $b = b'$ *and* $\forall\, i \in [Q_{\mathsf{ct}}],\ j \in [Q_{\mathsf{sk}}], f^j(x^i_0) = f^j(x^i_1)$*,* 0 *otherwise.*

As for simulation security, we say that FE *satisfies subexponential security if* $\mathsf{negl}(\lambda) = 2^{-\lambda^{\Omega(1)}}$*.*

4.2 Efficiency Features

We now define various efficiency notions for PHFE (which are straightforward to adapt to FE).

Definition 5 (Linear efficiency).
We say a PHFE for the functionality $\mathcal{F} = \{\mathcal{F}_{\lambda,\mathsf{prmtr}}\}_{\lambda,\mathsf{prmtr}}$ *satisfies linear efficiency if there exists a polynomial* poly *such that for all security parameters* $\lambda \in \mathbb{N}$ *and all polynomial parameters* $\mathsf{prmtr} = (n, d, \ell, \mathsf{size})$*, all messages* $(x, y) \in \mathcal{X}_{\lambda,\mathsf{prmtr}} \times \mathcal{Y}_{\lambda,\mathsf{prmtr}}$*, all* pp *in the support of* $\mathsf{PPGen}(1^\lambda, \mathsf{prmtr})$*, all* $(\mathsf{pk}, \mathsf{msk})$ *in the support of* $\mathsf{Setup}(\mathsf{pp})$ *the size of the circuit computing* $\mathsf{Enc}(\mathsf{pk}, \cdot)$ *on the input* (x, y) *is at most* $(|x| + |y|) \cdot \mathsf{poly}(\lambda)$*, for some fixed polynomial* poly *where* $|x|$ *and* $|y|$ *denote the size of* x *and* y*, respectively.*

Now we define the notion of sublinearity for FE scheme for the functionality $\mathcal{F}$ (i.e. all polynomial circuits, defined in Section 3). It was shown in a series of works [13, 26, 24] that such FE schemes for $\mathsf{P/poly}$ imply obfuscation (assuming subexponential security).

Definition 6 (Sublinearity). *Let* FE *be an FE scheme for the functionality* $\mathcal{F} = \{\mathcal{F}_{\lambda,\mathsf{prmtr}}\}_{\lambda,\mathsf{prmtr}}$*. If there exists* $\epsilon \in (0,1)$ *and a polynomial* poly *such that for all tuple of polynomials* $\mathsf{prmtr} = (n, d, \ell, \mathsf{size})$*, all* $\lambda \in \mathbb{N}$*, all* pp *in the support of* $\mathsf{PPGen}(1^\lambda, \mathsf{prmtr})$*, all* $(\mathsf{pk}, \mathsf{msk})$ *in the support of* $\mathsf{Setup}(\mathsf{pp})$*:*

- *if the size of the circuit* $\mathsf{Enc}(\mathsf{pk}, \cdot)$ *is at most* $\mathsf{size}^{1-\epsilon} \cdot \mathsf{poly}(n, \lambda)$ *then* FE *is said to be sublinearly efficient. It is said to be compact if* $\epsilon = 1$*.*
- *if for all* $x \in \{0,1\}^n$*, all ciphertexts* ct *in the support of* $\mathsf{Enc}(\mathsf{pk}, x)$*, the size of* ct *is at most* $\mathsf{size}^{1-\epsilon} \cdot \mathsf{poly}(n, \lambda)$ *then* FE *is said to be sublinearly ciphertext-efficient.*
- *if for all* $x \in \{0,1\}^n$*, all ciphertexts* ct *in the support of* $\mathsf{Enc}(\mathsf{pk}, x)$*, the size of* ct *is at most* $\ell^{1-\epsilon} \cdot \mathsf{poly}(n, \lambda)$ *then* FE *is said to be sublinearly output-efficient.*

Remark 7 (levelled linear efficiency, compactness, and sublinearity). More generally, we say that the scheme satisfies levelled linear efficiency or levelled compactness, or levelled sublinearity if the multiplicative factor $\mathsf{poly}(n, \lambda)$ in Definition 5 or Definition 6 is replaced by $\mathsf{poly}(\lambda, n, d)$, i.e. the polynomial also depends on the depth bound d.

4.3 Structural Properties

Now we define some structural properties that are very specific to our construction. First we define the notion of special structure which captures the property of a function key can be generated just by applying a linear function of the master secret key over some field along with the fact that the decryption of a ciphertext is "almost linear" (specified below).

Definition 7. *(Special Structure*.) We say that a functional encryption scheme* FE *for* $\mathcal{F}^{\mathsf{CIRC}} = \{\mathcal{F}^{\mathsf{CIRC}}_{\lambda,\mathsf{prmtr}}\}_{\lambda,\mathsf{prmtr}}$ *satisfies special structure* if there exist polynomials* h_1, h_2, h_3, h_4 *such that the following holds. Recall* $\mathcal{F}^{\mathsf{CIRC}}_{\lambda,\mathsf{prmtr}}$ *for* $\mathsf{prmtr} = (n, d, \ell, \mathsf{size})$ *consists of all Boolean circuits with* n *bits of input,* ℓ *bits of output, depth* d *and size* size*.*

- *(PP Syntax.) The* pp *generated by the* $\mathsf{PPGen}(1^\lambda, \mathsf{prmtr})$ *algorithm contains a* $h_1(\lambda)$*-bit prime modulus* p*.*
- *(Linear secret key Structure.) The master secret key is a vector in* $\boldsymbol{s} \in \mathbb{Z}_p^{h_2(\lambda)}$*. For any function* $f \in \mathcal{F}_{\lambda,\mathsf{prmtr}}$*, let* $f = \{f_i\}_{i\in[\ell]}$ *denote the circuit computing* i^{th} *bit of* f*. The functional secret key is of the form* $\mathsf{sk}_f = \{\mathsf{sk}_{f_i}\}_{i\in[\ell]}$ *where each* $\mathsf{sk}_{f_i} = \langle \mathsf{pp}_{f_i}, \boldsymbol{s}\rangle + e_i \mod p$ *where* $e_i \leftarrow_R \{0, \ldots, h_3(\lambda, n, \ell, d)\}$ *and* pp_{f_i} *is some deterministic polynomial time computable function of* pp *and* f_i*.*
- *(Linear + Round Decryption with polynomial decryption error.) There exists a deterministic poly-time algorithm such that given an encryption* ct *of* $m \in \{0,1\}^n$ *and a function* $f = (f_1, \ldots, f_\ell) \in \mathcal{F}_{\lambda,\mathsf{prmtr}}$*, for every* $i \in [\ell]$*, computes* ct_{f_i} *such that* $|\mathsf{ct}_{f_i} - \langle \mathsf{pp}_{f_i}, \boldsymbol{s}\rangle - f_i(m)\lceil \frac{p}{2} \rceil| \leq h_4(\lambda, d, \ell, \mathsf{size})$*. Given the secret-key for a function* $f = (f_1, \ldots, f_\ell)$*, this can be used to recover* $f(m) = (f_1(m), \ldots, f_\ell(m))$*.*

Outline *In the rest of the paper, we just discuss one of the aspect, which is to construct an from a* PHFE *scheme, an* ϵ-1LGFE *scheme and an* sPRG *an* ϵ*-secure Functional Encryption scheme. We show how to construct each of these primitives in the full version [45]. We also show in the full version how to amplify its security resulting into a sublinearly efficient Functional Encryption scheme. Such a scheme can be used to build* $i\mathcal{O}$ *using known results [13, 26].*

5 Definition of Structured-Seed PRG

We recall the notion of a structured seed PRG sPRG [58].

Definition 8 (Syntax of Structured-Seed Pseudo-Random Generators (sPRG)). *Let* τ *be a positive constant. A structured-seed Boolean PRG,* sPRG*, with stretch* τ *that maps* $(n \cdot \mathsf{poly}(\lambda))$*-bit binary strings into* $(m = n^\tau)$*-bit strings, where* poly *is a fixed polynomial, is defined by the following PPT algorithms:*

- $\mathsf{PPGen}(1^\lambda, 1^n)$ *takes as input the security parameter* λ*, and an input length* 1^n*, which is a polynomial in* λ*. It outputs public parameters* pp*, which amongst other things contains an odd prime modulus* $p(\lambda)$ *which is* $\mathsf{poly}(\lambda)$ *bit prime for some polynomial independent of* n*.*
- $\mathsf{IdSamp}(\mathsf{pp})$ *samples a function index* I*.*
- $\mathsf{SdSamp}(I)$ *jointly samples two binary strings, a public seed and a private seed,* $\mathsf{sd} = (\mathsf{P}, \mathsf{S})$*. These are vectors over* $\mathbb{Z}_p$*. The combined dimension of these vectors is* $n \cdot \mathsf{poly}(\lambda)$*.*
- $\mathsf{Eval}(I, \mathsf{sd})$ *computes a string in* $\{0,1\}^m$*.*

Remark 8 (The modulus $p(\lambda)$*).* The size of the modulus $p(\lambda)$ is some fixed polynomial in the security parameter λ independent of n.

Remark 9 (Polynomial Stretch.). We say that an sPRG has polynomial stretch if $\tau > 1$ for some constant τ.

Remark 10 (Linear Efficiency.). We say that an sPRG has linear-efficiency if the time to sample sd is $n \cdot \mathsf{poly}(\lambda)$.

Remark 11 (On $\mathsf{poly}(\lambda)$ *multiplicative factor in the seed length.).* As opposed to a standard Boolean PRG definition where the length of the output is set to be n^τ where n is the seed length, we allow the length of the seed to increase multiplicatively by a fixed polynomial poly in a parameter λ. Looking ahead, one should view n as an arbitrary large polynomial in λ, and hence sPRG will be expanding in length.

Definition 9 (Security of sPRG). *A structured-seed Boolean PRG,* sPRG*, satisfies*

Pseudorandomness: *Let* $\lambda \in \mathbb{N}$ *be the security parameter, let* $n(\lambda)$ *be a polynomial in* λ*. Then, following distributions are indistinguishable.*

$$(\mathsf{pp}, I, \mathsf{P}, \mathsf{Eval}(I, \mathsf{sd}))$$
$$(\mathsf{pp}, I, \mathsf{P}, \boldsymbol{r})$$

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

Definition 10 (Complexity and degree of sPRG). *Let $D \in \mathbb{N}$, let $\lambda \in \mathbb{N}$ and $n = n(\lambda)$ be arbitrary positive polynomial in λ, and $p = p(\lambda)$ denote a prime modulus which is sampled during* PPGen*. Let $\mathbb{C}$ be a complexity class. A* sPRG *has complexity $\mathbb{C}$ in the public seed and degree D in private seed over $\mathbb{Z}_p$, denoted as,* $\mathsf{sPRG} \in (\mathbb{C},\ deg\ D)$*, if for every I in the support of* $\mathsf{IdSamp}(1^\lambda, 1^n)$*, there exists an algorithm* $\mathsf{Process}_I$ *in $\mathbb{C}$ and an $m(n)$-tuple of polynomials Q_I that can be efficiently generated from I, such that for all* sd *in the support of* $\mathsf{SdSamp}(I)$*, it holds that:*

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

where Q_I has degree 1 in P *and degree D in* S*.*

We remark that the above definition generalizes the standard notion of families of PRGs in two aspects: 1) the seed consists of a public part and a private part, jointly sampled and arbitrarily correlated, and 2) the seed may not be uniform. In the full version, we show how to construct an sPRG from our new assumption $\mathsf{G\text{-}LWEleak}_{D,\epsilon,\rho}$.

6 Construction of ϵ-Simulation Secure FE

In this section, we construct a ϵ-simulation secure public-key functional encryption scheme FE for circuits $\mathcal{F}^{\mathsf{CIRC}} = \{\mathcal{F}^{\mathsf{CIRC}}_{\lambda,\mathsf{prmtr}}\}_{\lambda,\mathsf{prmtr}}$ for some $\epsilon \in (0,1)$. $\mathcal{F}^{\mathsf{CIRC}}_{\lambda,\mathsf{prmtr}}$ is the function class where for all λ and all polynomials $\mathsf{prmtr} = (n, d, \ell, \mathsf{size})$ it denotes the set of Boolean circuits with input length $n(\lambda)$, depth at most $d(\lambda)$, output length $\ell(\lambda)$, and size at most $\mathsf{size}(\lambda)$. It uses the following ingredients:

- ϵ-1LGFE: a secret-key FE scheme for the function class $\mathcal{F}^{\mathsf{CIRC}}$ defined above, satisfying the following properties:
  - (Security.) 1-key single ciphertext ϵ-simulation security as in Definition 4 for some constant $\epsilon \in (0,1)$ specified later. Note that although the scheme is for a single key, it however allows circuits with multiple output bits.
  - (Efficiency.) *levelled* compactness as in Definition 5. In particular, ciphertext size as well as the size of encryption circuit is $\mathsf{poly}(\lambda, n, d)$, independent of the function size size and output length ℓ.
  - (Structural property.) Special Structure* as per Definition 7. Recall, it says that:
    - (PP Syntax.) The pp generated by the $\mathsf{PPGen}(1^\lambda, \mathsf{prmtr})$ algorithm contains a $h_1(\lambda)$-bit prime modulus which is the modulus of the bilinear map $\mathcal{G}_\lambda$, p.
    - (Linear secret key Structure.) The master secret key is a vector in $\boldsymbol{s} \in \mathbb{Z}_p^{h_2(\lambda)}$. For any function $f \in \mathcal{F}_{\lambda,\mathsf{prmtr}}$, let $f = \{f_i\}_{i\in[\ell]}$ denote the

circuit computing i^{th} bit of f. The functional secret key is of the form $\mathsf{sk}_f = \{\mathsf{sk}_{f_i}\}_{i\in[\ell]}$ where each $\mathsf{sk}_{f_i} = \langle \mathsf{pp}_{f_i}, \boldsymbol{s}\rangle + e_i \mod p$ where $e_i \leftarrow_{\mathrm{R}} \{0,\ldots,h_3(\lambda,n,\ell,d)\}$ and pp_{f_i} is some deterministic polynomial time computable function of pp and f_i. For our construction below we require that $h_3(\lambda,n,\ell,d) = 2^t - 1$ for some natural number $t = O(\log(n\cdot d\cdot \ell\cdot \mathsf{size}))$. We can always choose an a constant $\epsilon \in (0,1)$ for the construction in the full version [45] such that there exists an ϵ-1LGFE scheme with this property, satisfying ϵ-simulation security. We use that value of ϵ.

  - $*$ (Linear + Round Decryption with polynomial decryption error.) There exists a deterministic poly-time algorithm such that given an encryption ct of $m \in \{0,1\}^n$ and a function $f = (f_1,\ldots,f_\ell) \in \mathcal{F}^{\mathsf{CIRC}}_{\lambda,\mathsf{prmtr}}$, for every $i\in[\ell]$, computes ct_{f_i} such that $|\mathsf{ct}_{f_i} - \langle \mathsf{pp}_{f_i}, \boldsymbol{s}\rangle - f_i(m)\lceil \frac{p}{2}\rceil| \le h_4(\lambda,d,\ell,\mathsf{size})$. Given the secret-key for a function $f = (f_1,\ldots,f_\ell)$, this can be used to recover $f(m) = (f_1(m),\ldots,f_\ell(m))$.

  Such a scheme is constructed in the full version [45].
- PHFE: a public-key PHFE for the class of functions $\mathcal{F}^{\mathsf{PHFE}}$ defined with respect to bilinear groups of order p (which is the same as the modulus of ϵ-1LGFE) and is in fact the order of group $\mathcal{G}_\lambda$. $\mathcal{F}^{\mathsf{PHFE}} = \{\mathcal{F}^{\mathsf{PHFE}}_{\lambda,n'}\}_{\lambda,n'}$ for every polynomial n' consists of all functions f that takes an input of the form $(\boldsymbol{x},\boldsymbol{y}) \in \mathbb{Z}_p^{n'} \times \mathbb{Z}_p^{n'}$, and computes $f(\boldsymbol{x},\boldsymbol{y}) = [\sum_{j,k} f_{j,k}(\boldsymbol{x})\cdot y_j\cdot y_k]_T \in \mathbf{G}_T$ where $f_{j,k}$ is a constant degree polynomial over $\boldsymbol{x}$ (i.e. an arithmetic circuit in NC^0), and $\mathbf{G}_T$ denotes the target group (see def pairings). The scheme PHFE satisfies the following properties:
  - (Security.) 1-simulation security for unbounded key queries.
  - (Efficiency.) Linear run-time as per Definition 5.

  Such a scheme is constructed in the full version. We set n' later.
- sPRG: a structured-seed PRG with stretch $\tau > 1$, linear efficiency as per Definition 8. This sPRG works with the modulus $p(\lambda)$ of the bilinear map $\mathcal{G}_\lambda$. The evaluation algorithm of sPRG computes an arithmetic NC^0 circuit on the public part of the seed, and a degree-2 polynomial on the secret part of the seed, that is, $\mathsf{sPRG} \in (\mathsf{arith}\text{-}\mathsf{NC}^0, \deg\ 2)$. This sPRG is implementable by $\mathcal{F}^{\mathsf{PHFE}}$.

We now describe the construction.

Parameters: For sPRG, we set the length parameter to be $\ell^{\frac{1}{\tau}}\cdot\lambda$. Thus, $\ell_{\mathsf{sPRG}} = \ell^{\frac{1}{\tau}}\,\mathsf{poly}(\lambda)$ is the number of $\mathbb{Z}_p$ elements in the sPRG seed for some polynomial poly independent of the ℓ. Define $n' = h_2(\lambda,d) + \ell_{\mathsf{sPRG}}$. Let $t = \log_2(h_3(\lambda,n,\ell,d)+1)$.

Construction: Please refer to the construction in Figure 6.

Due to lack of space, we argue correctness, efficiency and security properties in the full version [45].

$\underline{\mathsf{FE.PPGen}(1^\lambda, \mathsf{prmtr}) :}$
Given 1^λ and the tuple of polynomials $\mathsf{prmtr} = (n, \mathsf{size}, d, \ell)$, it samples $\mathsf{PHFE.pp} \leftarrow \mathsf{PHFE.PPGen}(1^\lambda, 1^{n'})$, $\epsilon\text{-1LGFE.pp} \leftarrow \epsilon\text{-1LGFE.PPGen}(1^\lambda, \mathsf{prmtr})$ and $\mathsf{sPRG.pp} \leftarrow \mathsf{sPRG.PPGen}(1^\lambda, 1^{\ell^{\frac{1}{\tau}} \cdot \lambda})$, $I \leftarrow \mathsf{sPRG.IdSamp}(\mathsf{sPRG.pp})$. Let p denote the prime modulus of $\mathcal{G}_\lambda$. Output $\mathsf{pp} = (\mathsf{PHFE.pp}, \epsilon\text{-1LGFE.pp}, \mathsf{sPRG.pp}, I, p)$.

$\underline{\mathsf{FE.Setup}(\mathsf{pp}) :}$ Run $\mathsf{PHFE.Setup}(\mathsf{PHFE.pp}) \rightarrow (\mathsf{PHFE.pk}, \mathsf{PHFE.msk})$. Set and output $\mathsf{FE.pk} = \mathsf{PHFE.pk}$ and $\mathsf{FE.msk} = \mathsf{PHFE.msk}$.

$\underline{\mathsf{FE.Enc}(\mathsf{FE.pk}, m \in \{0,1\}^n) :}$

- $\mathsf{msk}' \leftarrow \epsilon\text{-1LGFE.Setup}(\epsilon\text{-1LGFE.pp})$
- $\mathsf{ct}_1 \leftarrow \epsilon\text{-1LGFE.Enc}(\mathsf{msk}', m)$.
- $(\mathsf{P}, \mathsf{S}) \leftarrow \mathsf{SdSamp}(I)$.
- $\mathsf{ct}_2 \leftarrow \mathsf{PHFE.Enc}(\mathsf{PHFE.pk}, (\mathsf{P}, (\mathsf{S}, \mathsf{msk}')))$.

It returns $\mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2)$.

$\underline{\mathsf{FE.KeyGen}(\mathsf{FE.msk}, C) :}$ Given as input a circuit $C \in \mathcal{F}_{\mathsf{prmtr}}$, denote $C = (C_1, \ldots, C_\ell)$ where each C_i is the circuit computing the i^{th} output bit of C. For every $i \in [\ell]$, do the following:

- let $\epsilon\text{-1LGFE.pp}_{C_i}$ be the vector computed deterministically from $\epsilon\text{-1LGFE.pp}$ and C_i such that $\mathsf{sk}_{C_i} \approx \langle \mathsf{msk}', \epsilon\text{-1LGFE.pp}_{C_i} \rangle$ (see the linear secret key structure in Definition 7).
- Compute $\mathsf{sk}_{C_i} \leftarrow \mathsf{PHFE.KeyGen}(\mathsf{PHFE.msk}, f_i)$ where f_i takes as input $(\mathsf{P}, (\mathsf{S}, \mathsf{msk}'))$ and outputs $\langle \mathsf{msk}', \epsilon\text{-1LGFE.pp}_{C_i} \rangle + \sum_{j \in [1,t]} 2^{j-1} \cdot r_{(i-1)\cdot t + j}$, where for all $\theta \in [m]$, r_θ denotes the θ'th bit output by $\mathsf{sPRG.Eval}(I, \mathsf{sd}) \in \{0,1\}^m$.

It returns $\mathsf{sk}_C = (\mathsf{sk}_{C_1}, \ldots \mathsf{sk}_{C_\ell})$.

$\underline{\mathsf{FE.Dec}(\mathsf{sk}_C, \mathsf{ct}) :}$ Parse $\mathsf{sk}_C = (\mathsf{sk}_{C_1}, ..., \mathsf{sk}_{C_\ell})$ and $\mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2)$. For every $i \in [\ell]$, do the following:

- By the special structure* of ϵ-1LGFE, compute $\mathsf{ct}_{C,i}$ using the ciphertext ct_1.
- Compute $[w_i]_T \leftarrow \mathsf{PHFE.Dec}(\mathsf{sk}_{C_i}, \mathsf{ct}_2)$.
- Compute $[z_i]_T = [\mathsf{ct}_{C_i} - w_i]_T$.
- Check if $|z_i| \leq h_3(\lambda, n, d, \ell) + h_4(\lambda, n, d, \ell)$ (by brute-force). If so set $y_i = 0$. Otherwise, set $y_i = 1$. Output $(y_1, ..., y_\ell)$.

Fig. 1. Construction of Functional Encryption Scheme FE.

7 Acknowledgements

Aayush Jain was partially supported by grants listed under Amit Sahai, a Google PhD fellowship. Huijia Lin was supported by NSF grants CNS-1528178, CNS-1929901, CNS-1936825 (CAREER), the Defense Advanced Research Projects Agency (DARPA) and Army Research Office (ARO) under Contract No. W911NF-15-C-0236, and a subcontract No. 2017-002 through Galois.

Amit Sahai was supported in part from DARPA SAFEWARE and SIEVE awards, NTT Research, NSF Frontier Award 1413955, and NSF grant 1619348, BSF grant 2012378, a Xerox Faculty Research Award, a Google Faculty Research Award, an equipment grant from Intel, and an Okawa Foundation Research Grant. This material is based upon work supported by the Defense Advanced Research Projects Agency through Award HR00112020024 and the ARL under Contract W911NF-15-C- 0205.

The views expressed are those of the authors and do not reflect the official policy or position of the Department of Defense, DARPA, ARO, Simons, Intel, Okawa Foundation, ODNI, IARPA, DIMACS, BSF, Xerox, the National Science Foundation, NTT Research, Google, or the U.S. Government.

References

1. Michel Abdalla, Florian Bourse, Angelo De Caro, and David Pointcheval. Simple functional encryption schemes for inner products. In Jonathan Katz, editor, *PKC 2015*, volume 9020 of *LNCS*, pages 733–751. Springer, Heidelberg, March / April 2015.
2. Michel Abdalla, Dario Catalano, Romain Gay, and Bogdan Ursu. Inner-product functional encryption with fine-grained access control. Cryptology ePrint Archive, Report 2020/577, 2020. `https://eprint.iacr.org/2020/577`.
3. Shweta Agrawal. Indistinguishability obfuscation without multilinear maps: New methods for bootstrapping and instantiation. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part I*, volume 11476 of *LNCS*, pages 191–225. Springer, Heidelberg, May 2019.
4. Shweta Agrawal, Benoit Libert, and Damien Stehle. Fully secure functional encryption for inner products, from standard assumptions. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part III*, volume 9816 of *LNCS*, pages 333–362. Springer, Heidelberg, August 2016.
5. Shweta Agrawal and Alice Pellet-Mary. Indistinguishability obfuscation without maps: Attacks and fixes for noisy linear FE. In Anne Canteaut and Yuval Ishai, editors, *Advances in Cryptology - EUROCRYPT 2020 - 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10-14, 2020, Proceedings, Part I*, volume 12105 of *Lecture Notes in Computer Science*, pages 110–140. Springer, 2020.
6. Shweta Agrawal and Alon Rosen. Functional encryption for bounded collusions, revisited. In *TCC*, pages 173–205, 2017.
7. Shweta Agrawal and Alon Rosen. Functional encryption for bounded collusions, revisited. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017, Part I*, volume 10677 of *LNCS*, pages 173–205. Springer, Heidelberg, November 2017.

8. Martin R. Albrecht, Carlos Cid, Jean-Charles Faugère, Robert Fitzpatrick, and Ludovic Perret. Algebraic algorithms for LWE problems. *ACM Commun. Comput. Algebra*, 49(2):62, 2015.
9. Prabhanjan Ananth, Zvika Brakerski, Gil Segev, and Vinod Vaikuntanathan. From selective to adaptive security in functional encryption. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 657–677. Springer, Heidelberg, August 2015.
10. Prabhanjan Ananth, Divya Gupta, Yuval Ishai, and Amit Sahai. Optimizing obfuscation: Avoiding Barrington's theorem. In *ACM CCS*, pages 646–658, 2014.
11. Prabhanjan Ananth, Aayush Jain, Huijia Lin, Christian Matt, and Amit Sahai. Indistinguishability obfuscation without multilinear maps: New paradigms via low degree weak pseudorandomness and security amplification. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part III*, volume 11694 of *LNCS*, pages 284–332. Springer, Heidelberg, August 2019.
12. Prabhanjan Ananth, Aayush Jain, and Amit Sahai. Indistinguishability obfuscation without multilinear maps: io from lwe, bilinear maps, and weak pseudorandomness. *IACR Cryptology ePrint Archive*, 2018:615, 2018.
13. Prabhanjan Ananth and Abhishek Jain. Indistinguishability obfuscation from compact functional encryption. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part I*, volume 9215 of *LNCS*, pages 308–326. Springer, Heidelberg, August 2015.
14. Prabhanjan Ananth and Amit Sahai. Projective arithmetic functional encryption and indistinguishability obfuscation from degree-5 multilinear maps. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part I*, volume 10210 of *LNCS*, pages 152–181. Springer, Heidelberg, April / May 2017.
15. Benny Applebaum. Pseudorandom generators with long stretch and low locality from random local one-way functions. In Howard J. Karloff and Toniann Pitassi, editors, *44th ACM STOC*, pages 805–816. ACM Press, May 2012.
16. Benny Applebaum, Andrej Bogdanov, and Alon Rosen. A dichotomy for local small-bias generators. In Ronald Cramer, editor, *TCC 2012*, volume 7194 of *LNCS*, pages 600–617. Springer, Heidelberg, March 2012.
17. Benny Applebaum and Shachar Lovett. Algebraic attacks against random local functions and their countermeasures. In Daniel Wichs and Yishay Mansour, editors, *48th ACM STOC*, pages 1087–1100. ACM Press, June 2016.
18. Sanjeev Arora and Rong Ge. New algorithms for learning in presence of errors. In Luca Aceto, Monika Henzinger, and Jiri Sgall, editors, *ICALP 2011, Part I*, volume 6755 of *LNCS*, pages 403–415. Springer, Heidelberg, July 2011.
19. Saikrishna Badrinarayanan, Eric Miles, Amit Sahai, and Mark Zhandry. Post-zeroizing obfuscation: New mathematical tools, and the case of evasive circuits. In *Advances in Cryptology - EUROCRYPT*, pages 764–791, 2016.
20. 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, *CRYPTO 2017, Part I*, volume 10401 of *LNCS*, pages 67–98. Springer, Heidelberg, August 2017.
21. Boaz Barak, Sanjam Garg, Yael Tauman Kalai, Omer Paneth, and Amit Sahai. Protecting obfuscation against algebraic attacks. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 221–238. Springer, Heidelberg, May 2014.
22. Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In

Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 1–18. Springer, Heidelberg, August 2001.
23. James Bartusek, Yuval Ishai, Aayush Jain, Fermi Ma, Amit Sahai, and Mark Zhandry. Affine determinant programs: A framework for obfuscation and witness encryption. In Thomas Vidick, editor, *11th Innovations in Theoretical Computer Science Conference, ITCS 2020, January 12-14, 2020, Seattle, Washington, USA*, volume 151 of *LIPIcs*, pages 82:1–82:39. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.
24. Nir Bitansky, Ryo Nishimaki, Alain Passelègue, and Daniel Wichs. From cryptomania to obfustopia through secret-key functional encryption. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B, Part II*, volume 9986 of *LNCS*, pages 391–418. Springer, Heidelberg, October / November 2016.
25. Nir Bitansky, Omer Paneth, and Alon Rosen. On the cryptographic hardness of finding a Nash equilibrium. In Venkatesan Guruswami, editor, *56th FOCS*, pages 1480–1498. IEEE Computer Society Press, October 2015.
26. Nir Bitansky and Vinod Vaikuntanathan. Indistinguishability obfuscation from functional encryption. In Venkatesan Guruswami, editor, *56th FOCS*, pages 171–190. IEEE Computer Society Press, October 2015.
27. Andrej Bogdanov and Youming Qiao. On the security of goldreich's one-way function. *Comput. Complex.*, 21(1):83–127, 2012.
28. Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil pairing. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 213–229. Springer, Heidelberg, August 2001.
29. Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In Yuval Ishai, editor, *TCC 2011*, volume 6597 of *LNCS*, pages 253–273. Springer, Heidelberg, March 2011.
30. Zvika Brakerski, Nico Dottling, Sanjam Garg, and Guilio Malavolta. Candidate io from homomorphic encryption schemes. In *EUROCRYPT*, 2020.
31. Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. In *Innovations in Theoretical Computer Science 2012, Cambridge, MA, USA, January 8-10, 2012*, pages 309–325, 2012.
32. Zvika Brakerski and Guy N. Rothblum. Virtual black-box obfuscation for all circuits via generic graded encoding. In *TCC*, pages 1–25, 2014.
33. Zvika Brakerski and Vinod Vaikuntanathan. Fully homomorphic encryption from ring-LWE and security for key dependent messages. In Phillip Rogaway, editor, *CRYPTO 2011*, volume 6841 of *LNCS*, pages 505–524. Springer, Heidelberg, August 2011.
34. Christina Brzuska, Pooya Farshim, and Arno Mittelbach. Indistinguishability obfuscation and UCEs: The case of computationally unpredictable sources. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 188–205. Springer, Heidelberg, August 2014.
35. Johannes A. Buchmann, Florian Göpfert, Rachel Player, and Thomas Wunderer. On the hardness of LWE with binary error: Revisiting the hybrid lattice-reduction and meet-in-the-middle attack. In David Pointcheval, Abderrahmane Nitaj, and Tajjeeddine Rachidi, editors, *AFRICACRYPT 16*, volume 9646 of *LNCS*, pages 24–43. Springer, Heidelberg, April 2016.
36. Sun Caho, Mehdi Tibouchi, and Masayuki Abe. Sample-time trade-off for the arora-ge attack on binary lwe. *Symposium on Cryptography and Information Theory*, 2019.

37. Mehdi Tibouchi Chao Sun and Masayuki Abe. Revisiting the hardness of binary error lwe. Cryptology ePrint Archive, Report 2020/666, 2020. https://eprint.iacr.org/2020/666.
38. Yilei Chen, Vinod Vaikuntanathan, and Hoeteck Wee. GGH15 beyond permutation branching programs: Proofs, attacks, and candidates. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 577–607. Springer, Heidelberg, August 2018.
39. Aloni Cohen, Justin Holmgren, Ryo Nishimaki, Vinod Vaikuntanathan, and Daniel Wichs. Watermarking cryptographic capabilities. In *STOC*, 2016.
40. Geoffroy Couteau, Aurélien Dupin, Pierrick Méaux, Mélissa Rossi, and Yann Rotella. On the concrete security of Goldreich's pseudorandom generator. In Thomas Peyrin and Steven Galbraith, editors, *ASIACRYPT 2018, Part II*, volume 11273 of *LNCS*, pages 96–124. Springer, Heidelberg, December 2018.
41. Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *54th FOCS*, pages 40–49. IEEE Computer Society Press, October 2013.
42. Sanjam Garg, Eric Miles, Pratyay Mukherjee, Amit Sahai, Akshayaram Srinivasan, and Mark Zhandry. Secure obfuscation in a weak multilinear map model. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B, Part II*, volume 9986 of *LNCS*, pages 241–268. Springer, Heidelberg, October / November 2016.
43. Sanjam Garg, Omkant Pandey, and Akshayaram Srinivasan. Revisiting the cryptographic hardness of finding a nash equilibrium. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part II*, volume 9815 of *LNCS*, pages 579–604. Springer, Heidelberg, August 2016.
44. Romain Gay. A new paradigm for public-key functional encryption for degree-2 polynomials. In *PKC 2020, Part I*, LNCS, pages 95–120. Springer, Heidelberg, 2020.
45. Romain Gay, Aayush Jain, Huijia Lin, and Amit Sahai. Indistinguishability obfuscation from simple-to-state hard problems: New assumptions, new techniques, and simplification. *IACR Cryptol. ePrint Arch.*, 2020:764, 2020.
46. Craig Gentry, Charanjit S. Jutla, and Daniel Kane. Obfuscation using tensor products. *Electronic Colloquium on Computational Complexity (ECCC)*, 25:149, 2018.
47. Craig Gentry, Allison B. Lewko, Amit Sahai, and Brent Waters. Indistinguishability obfuscation from the multilinear subgroup elimination assumption. *IACR Cryptology ePrint Archive*, 2014:309, 2014.
48. Oded Goldreich. Candidate one-way functions based on expander graphs. *Electronic Colloquium on Computational Complexity (ECCC)*, 7(90), 2000.
49. Shafi Goldwasser, S. Dov Gordon, Vipul Goyal, Abhishek Jain, Jonathan Katz, Feng-Hao Liu, Amit Sahai, Elaine Shi, and Hong-Sheng Zhou. Multi-input functional encryption. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 578–602. Springer, Heidelberg, May 2014.
50. Shafi Goldwasser, Yael Tauman Kalai, Raluca A. Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich. Reusable garbled circuits and succinct functional encryption. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th ACM STOC*, pages 555–564. ACM Press, June 2013.
51. Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. One-time programs. In David Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 39–56. Springer, Heidelberg, August 2008.

52. Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Functional encryption with bounded collusions via multi-party computation. In *Advances in Cryptology - CRYPTO 2012 - 32nd Annual Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2012. Proceedings*, pages 162–179, 2012.
53. Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Functional encryption with bounded collusions via multi-party computation. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 162–179. Springer, Heidelberg, August 2012.
54. Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Predicate encryption for circuits from LWE. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 503–523. Springer, Heidelberg, August 2015.
55. Dennis Hofheinz, Tibor Jager, Dakshita Khurana, Amit Sahai, Brent Waters, and Mark Zhandry. How to generate and use universal samplers. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part II*, volume 10032 of *LNCS*, pages 715–744. Springer, Heidelberg, December 2016.
56. Susan Hohenberger, Amit Sahai, and Brent Waters. Full domain hash from (leveled) multilinear maps and identity-based aggregate signatures. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 494–512. Springer, Heidelberg, August 2013.
57. Aayush Jain, Huijia Lin, Christian Matt, and Amit Sahai. How to leverage hardness of constant-degree expanding polynomials overa $\mathbb{R}$ to build $i\mathcal{O}$. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part I*, volume 11476 of *LNCS*, pages 251–281. Springer, Heidelberg, May 2019.
58. Aayush Jain, Huijia Lin, and Amit Sahai. Indistinguishability obfuscation from well-founded assumptions. Cryptology ePrint Archive, Report 2020/1003, 2020. `https://eprint.iacr.org/2020/1003`.
59. Antoine Joux. A one round protocol for tripartite diffie-hellman. In Wieb Bosma, editor, *Algorithmic Number Theory, 4th International Symposium, ANTS-IV, Leiden, The Netherlands, July 2-7, 2000, Proceedings*, volume 1838 of *Lecture Notes in Computer Science*, pages 385–394. Springer, 2000.
60. Venkata Koppula, Allison Bishop Lewko, and Brent Waters. Indistinguishability obfuscation for turing machines with unbounded memory. In *STOC*, 2015.
61. Pravesh K. Kothari, Ryuhei Mori, Ryan O'Donnell, and David Witmer. Sum of squares lower bounds for refuting any CSP. In Hamed Hatami, Pierre McKenzie, and Valerie King, editors, *49th ACM STOC*, pages 132–145. ACM Press, June 2017.
62. Huijia Lin. Indistinguishability obfuscation from constant-degree graded encoding schemes. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part I*, volume 9665 of *LNCS*, pages 28–57. Springer, Heidelberg, May 2016.
63. Huijia Lin. Indistinguishability obfuscation from SXDH on 5-linear maps and locality-5 PRGs. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part I*, volume 10401 of *LNCS*, pages 599–629. Springer, Heidelberg, August 2017.
64. Huijia Lin and Christian Matt. Pseudo flawed-smudging generators and their application to indistinguishability obfuscation. *IACR Cryptology ePrint Archive*, 2018:646, 2018.
65. Huijia Lin and Stefano Tessaro. Indistinguishability obfuscation from trilinear maps and block-wise local PRGs. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part I*, volume 10401 of *LNCS*, pages 630–660. Springer, Heidelberg, August 2017.

66. Huijia Lin and Vinod Vaikuntanathan. Indistinguishability obfuscation from DDH-like assumptions on constant-degree graded encodings. In Irit Dinur, editor, *57th FOCS*, pages 11–20. IEEE Computer Society Press, October 2016.
67. Alex Lombardi and Vinod Vaikuntanathan. Minimizing the complexity of goldreich's pseudorandom generator. *IACR Cryptology ePrint Archive*, 2017:277, 2017.
68. Daniele Micciancio and Chris Peikert. Hardness of SIS and LWE with small parameters. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 21–39. Springer, Heidelberg, August 2013.
69. Elchanan Mossel, Amir Shpilka, and Luca Trevisan. On e-biased generators in NC0. In *44th FOCS*, pages 136–145. IEEE Computer Society Press, October 2003.
70. Ryan O'Donnell and David Witmer. Goldreich's PRG: evidence for near-optimal polynomial stretch. In *IEEE 29th Conference on Computational Complexity, CCC 2014, Vancouver, BC, Canada, June 11-13, 2014*, pages 1–12. IEEE Computer Society, 2014.
71. Adam O'Neill. Definitional issues in functional encryption. *IACR Cryptology ePrint Archive*, 2010:556, 2010.
72. Rafael Pass, Karn Seth, and Sidharth Telang. Indistinguishability obfuscation from semantically-secure multilinear encodings. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 500–517. Springer, Heidelberg, August 2014.
73. Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In *STOC*, pages 84–93, 2005.
74. Amit Sahai and Hakan Seyalioglu. Worry-free encryption: functional encryption with public keys. In *Proceedings of the 17th ACM conference on Computer and communications security*, pages 463–472. ACM, 2010.
75. Amit Sahai and Hakan Seyalioglu. Worry-free encryption: functional encryption with public keys. In Ehab Al-Shaer, Angelos D. Keromytis, and Vitaly Shmatikov, editors, *ACM CCS 2010*, pages 463–472. ACM Press, October 2010.
76. Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In David B. Shmoys, editor, *STOC*, pages 475–484. ACM, 2014.
77. Amit Sahai and Brent R. Waters. Fuzzy identity-based encryption. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 457–473. Springer, Heidelberg, May 2005.